

CYBER 170 State
Hardware Reference

REFERENCE
Guide

 / CYBER 960
Computer System

12-30-88

JPS

CYBER 960 Computer System

CYBER 170 State

Hardware Reference

CYBER 960 Computer System

CYBER 170 State

Hardware Reference

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features and parameters.

Manual History

Manual released at Revision A in September 1988.

Revision	Change Order	Date	Reason for Change
A	-	09-23-88	Manual released.

©1988 by Control Data Corporation
All rights reserved.
Printed in the United States of America.

Contents

About This Manual	5	Instruction Descriptions	2-1
Audience	5	CP Instruction Formats	2-1
Organization	5	Instruction Description	
FCC Compliance	5	Nomenclature.....	2-3
Conventions	5	CP Operating Modes	2-4
Disclaimer	5	CP Instruction Descriptions	2-4
Related Manuals	5	Programming Information	3-1
Additional Related Manuals	7	CP Programming	3-1
Ordering Manuals	7	Memory Programming	3-25
Submitting Comments	7	Glossary	A-1
System Description	1-1	Index	Index-1
Registers	1-2		

Figures

1-1. System Block Diagram	1-2	3-3. Floating-Add Result Format	3-11
1-2. CYBER 170 Exchange Package .	1-3	3-4. Multiply Result Format	3-11
2-1. CP Instruction Parcel		3-5. Format of Exit Condition	
Arrangement.....	2-2	Register at RAC.....	3-16
3-1. CYBER 170 Exchange Package .	3-2	3-6. Memory Map	3-26
3-2. FP Format	3-5	3-7. CM Layout	3-27

Tables

2-1. CP Instruction Designators	2-3	3-1. Bits 58 and 59 Configurations ..	3-5
2-2. CP Integer Arithmetic		3-2. Xj Plus Xk (30, 32, and 34	
Instructions	2-5	Instructions).....	3-8
2-3. CP Branch Instructions	2-9	3-3. Xj Minus Xk (31, 33, and 35	
2-4. CP Block Copy Instructions	2-16	Instructions).....	3-9
2-5. CP Shift Instructions	2-19	3-4. Xj Multiplied by Xk (40, 41,	
2-6. CP Logical Instructions	2-23	and 42 Instructions).....	3-9
2-7. CP FP Instructions	2-27	3-5. Xj Divided by Xk (44 and 45	
2-8. CP Jump Instructions	2-39	Instructions).....	3-10
2-9. CP Exchange Jump		3-6. Contents of Exit Condition	
Instructions	2-41	Register at RAC.....	3-16
2-10. CP Compare/Move		3-7. Error Exits in CYBER 170	
Instructions	2-44	Monitor Mode (MF=1).....	3-17
2-11. Collate Table	2-48	3-8. Error Exits in CYBER 170 Job	
		Mode (MF=0).....	3-20

About This Manual

This manual contains hardware reference information for the CDC® CYBER 960 Computer Systems. This manual includes descriptions of functional characteristics and instructions unique to CYBER 170 State.

Audience

This manual is for use by customer, marketing, training, programming, and Engineering Services personnel who operate, program, and maintain the CYBER 960 computer systems.

Organization

The manual describes the functional, operational, and programming characteristics of the CYBER 170 State central processor (CP). Additional hardware reference information is available in the hardware manuals figure.

Chapter 1 contains the system description. Chapter 2 contains instruction descriptions, and chapter 3 contains programming information applicable to all the CYBER 170 computer systems. Appendix A contains the glossary. Command quick-reference pages can be found at the back of this manual.

FCC Compliance

This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device (insert peripheral computing device if appropriate) pursuant to Subpart J of Part 15 of the FCC Rules which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Conventions

New features, as well as technical changes, deletions, and additions to this manual, are indicated by vertical bars in the margins.

Disclaimer

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features and parameters.

Related Manuals

Additional hardware reference information regarding operation of the computer systems in both their CYBER 170 State and Virtual State environments is available in manuals shown in the hardware manuals figure on the following page.

CYBER 960/962 HARDWARE MANUALS

SYSTEM MANUALS

Site Preparation

General
Information
Site Preparation

60275100

Mainframe
Complex Data
Site Preparation

60000119

Peripheral
Equipment Data
Site Preparation

60275300

Installation

Mainframe
Installation and
Checkout

60000120

Operation

Virtual State
Reference

Volume 1

60000132

Virtual State
Reference

Volume 2

60000133

CYBER 170 State
Reference

60000127

19003 System
Console Operations/
Maintenance

60463610

Troubleshooting

System
Troubleshooting

60000122

Codes

System Codes
Booklet

60458100

EQUIPMENT MANUALS

Central Processing Unit (CPU)

CPU Maintenance

60000123

CPU Theory and
Diagrams

60000118

Mainframe Power and Environmental

Mainframe Power
and Environmental
Subsystem
Maintenance

60000125

Mainframe Power
and Environmental
Subsystem
Theory and
Diagrams

60000121

Input/Output Unit (IOU)

IOU Maintenance

60000130

CYBER 960 IOU Manuals

IOU Theory and
Diagrams

Levels 0-3

60463540

IOU Diagrams

Level 4

60462210

Wire Lists
Microfiche

60000134

CYBER 962 IOU Manuals

IOU Theory and
Diagrams

Levels 0-3

60000235

IOU Diagrams

Level 4

60000236

Wire Lists
Microfiche

60000135

Motor Generator (MG) Equipment

25-kVA Frequency
Converter
Hardware
Maintenance

60456520

40-kVA Control
Cabinet and Motor
Generator Hardware
Maintenance

60454720

MG Interface Unit
Installation and
Maintenance

60000124

M02240

Additional Related Manuals

Other manuals that are applicable to the CYBER 960 computer systems but not listed in the hardware manuals figure are:

Title	Publication Number
NOS Version 2 Operator/Analyst Handbook	60459310
NOS Version 2 Systems Programmer's Instant	60459370
NOS Version 1 Operator's Guide	60457700
NOS Version 1 Systems Programmer's Instant	60457790
CDC 721 Enhanced Display Terminal (CC634-B) Hardware Reference Manual	62950102
CYBER Initialization Package (CIP) Reference Manual	60457180

Publication ordering information and latest revision levels are available from the Literature Distribution and Services catalog, publication number 90310500.

Ordering Manuals

Control Data manuals are available through Control Data sales offices or through:

CONTROL DATA
Literature and Distribution Services
308 North Dale Street
St. Paul, Minnesota 55103-2495

The latest manual revision levels and ordering information are in the Literature and Distribution Services Catalog, publication number 90310500.

Submitting Comments

Control Data welcomes your comments about this manual. Your comments may include your opinion of the usefulness of this manual, your suggestions for specific improvements, and the reporting of any errors you have found.

You can submit your comments on the comment sheet on the last page of this manual. If the manual has no comment sheet, mail your comments to:

CONTROL DATA
Technical Publications ARH219
4201 Lexington Avenue North
St. Paul, Minnesota 55126-6198

Please indicate whether you would like a written response.

Registers	1-2
Operating Registers	1-4
X Registers	1-4
A Registers	1-4
B Registers	1-4
Support Registers	1-5
P Register	1-5
RAC Register	1-5
FLC Register	1-5
EM Register	1-6
Flag Register	1-6
RAE Register	1-7
FLE Register	1-7
MA Register	1-7

The CYBER 960 is capable of operating in either CYBER 170 or CYBER 180 (Virtual) State. The Virtual State Hardware Reference Manual Volumes 1 and 2 listed in About This Manual describe differences in IOU processors and channels as they relate to the different states.

This manual contains descriptions of the functional characteristics and instructions that are unique to CYBER 170 State.

The three models of the CYBER 960 CP have the following characteristics.

- Model 960-11 Base performance, Single CP
- Model 960-31 Enhanced performance, Single CP
- Model 960-32 Enhanced performance, Dual CP

In addition to the CP configurations, several IOU options are available to enhance IO processing. The base IOU for the 960 contains 20 nonconcurrent input/output (NIO) peripheral processors (PPs) and 24 C170 channels. An extension on the main cabinet (called an IOU expansion) can accommodate 5 or 10 concurrent input/output (CIO) PPs and 5 or 10 direct memory access (DMA) channels. The addition of a standalone IOU with a corresponding expansion cabinet makes available 10, 15, or 20 CIO PPs with 10, 15, or 20 DMA channels. DMA channels can operate in intelligent standard interface (ISI), intelligent peripheral interface (IPI), or C170 protocol.

The base central memory (CM) for the 960 contains 64M bytes of Random Access Memory (RAM). Upgrades allow the following memory sizes.

- 128M bytes
- 190M bytes
- 256M bytes

Registers

The CP contains the operating and support registers as described in the following paragraphs.

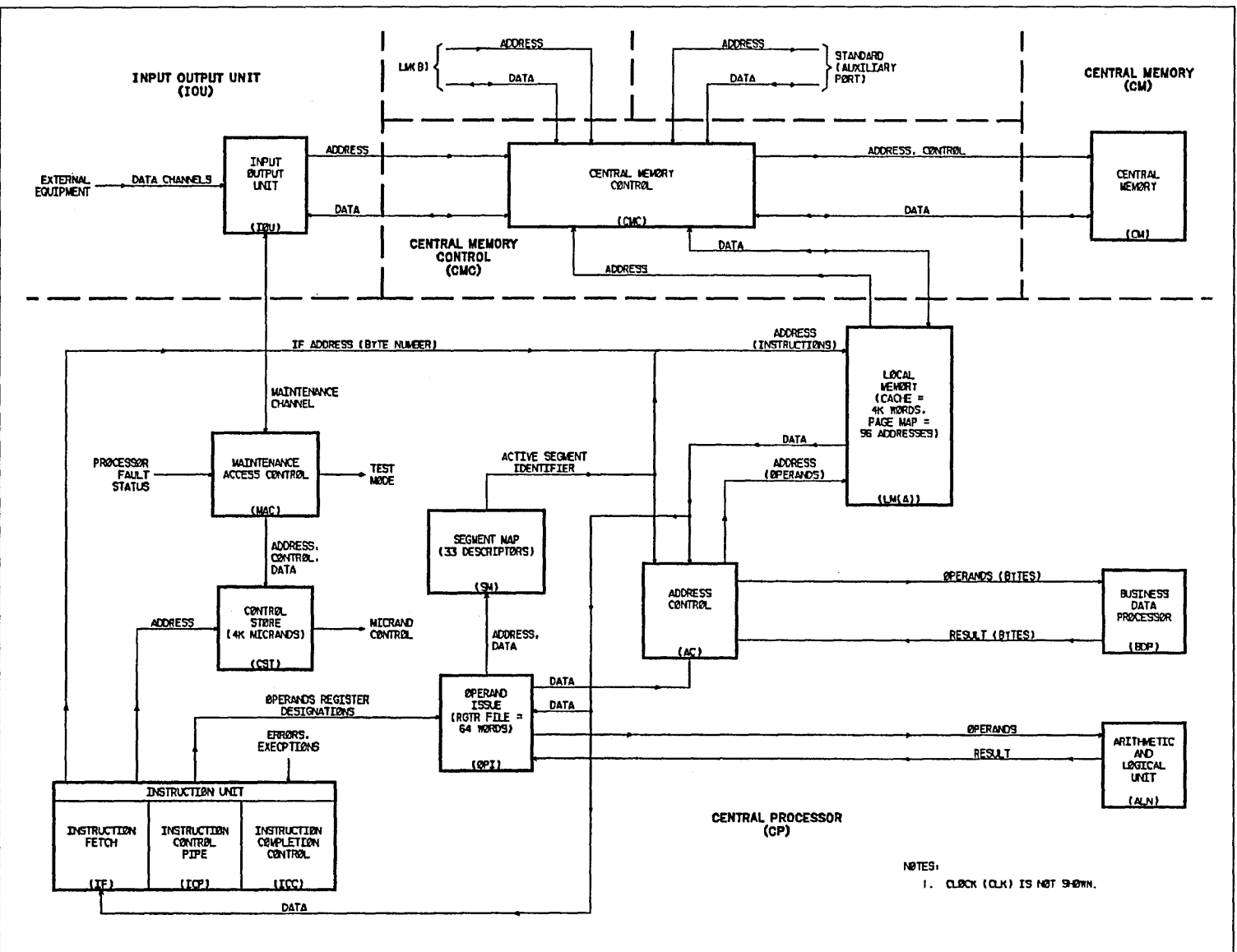


Figure 1-1. System Block Diagram

The contents of these registers can be written into memory and reloaded from memory as a CYBER 170 exchange package by a single CP instruction (CYBER 170 exchange jump). Figure 1-2 shows the CYBER 170 exchange package.

The time a CYBER 170 exchange package resides in CP hardware is called an execution interval. During this interval, the contents of X, A, B, and P registers can be changed by CP instructions. The contents of other support registers change only as a result of a CYBER 170 exchange jump. For further information, refer to CYBER 170 Exchange Jump in chapter 3.

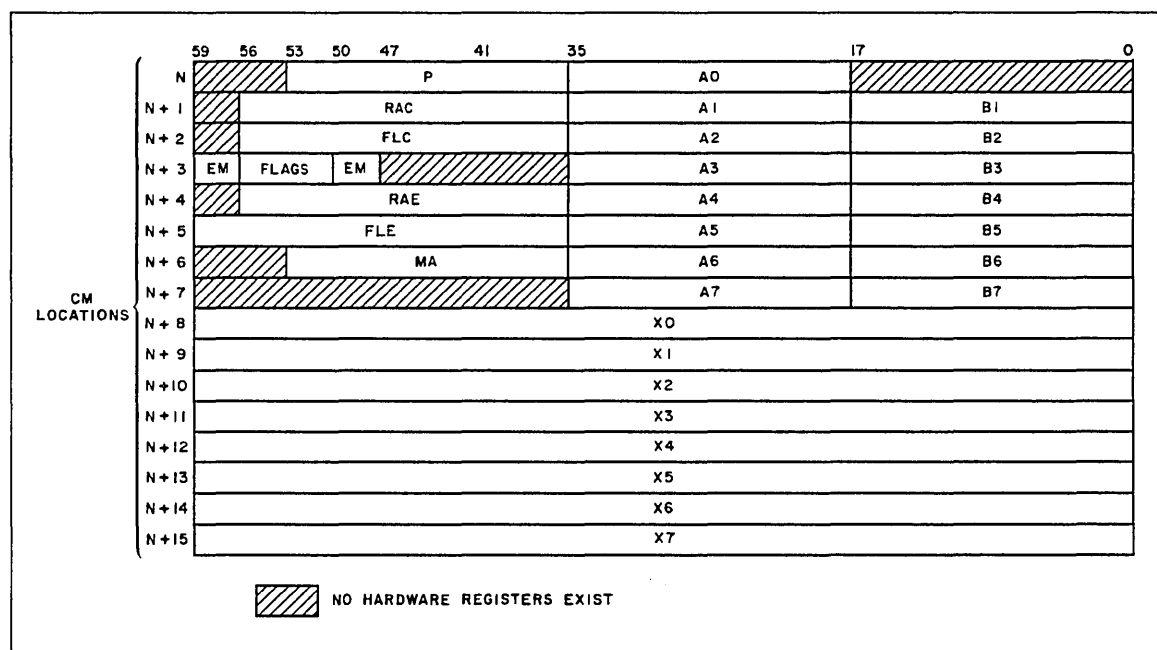


Figure 1-2. CYBER 170 Exchange Package

Operating Registers

The operating registers consist of X, A, and B registers. These registers minimize memory references for arithmetic operands and results.

X Registers

The CP contains eight 60-bit X registers, X0 through X7. The X0 register is used in the compare instructions to indicate if two fields of characters are equal. Also, the X0 register provides the relative Unified Extended Memory (UEM) starting address in a block-copy operation.

The X1 through X7 registers are primarily data handling registers for computation. X1 through X5 are used to input data from CM, and X6 and X7 are used to transmit data to CM.

Operands and results transfer between CM and the X registers as a result of placing CM addresses into corresponding A registers.

A Registers

The CP contains eight 18-bit A registers, A0 through A7. The A0 register serves as an intermediate register for the user's discretion. The A0 register is used in the compare-collate instruction for the collate table address. Also, the A0 register provides the relative CM starting address in a block-copy operation.

The registers A1 through A7 serve as CM operand address registers associated one-for-one with the X registers. Placing a quantity into an address register (A1 through A5) causes a CM read reference to that address and transmits the CM word to the corresponding X register (X1 through X5). Similarly, placing a quantity into the A6 or A7 register causes the word in the corresponding X6 or X7 register to be written into that relative address of CM.

B Registers

The CP contains eight 18-bit B registers, B0 through B7. These registers are primarily indexing registers to control program execution. Program loop counts may also be incremented or decremented in these registers.

Program addresses may be modified on the way to an A register by adding or subtracting B register quantities. The B registers also hold shift counts for the nominal B_j shifts, the resultant exponent for the unpack, the operand exponent for the pack, and the resultant shift count from a normalize. The B0 register always contains positive zero which can be used as an operand. This register cannot hold results from instructions.

Support Registers

Eight support registers assist the operating registers during the execution of programs. The contents of the support registers are stored in CM, and their new contents are loaded from CM during a CYBER 170 exchange sequence. With the exception of the P register, the contents of the support registers cannot be altered during the execution interval of a CYBER 170 exchange package. When the execution interval completes, the data in the support registers is sent back to CM through a CYBER 170 exchange jump.

P Register

The 18-bit program address (P) register loads from CM during the first word of a CYBER 170 exchange sequence and contains the current program execution address. The register serves as a program address counter and holds the relative CM address for each program step.

RAC Register

The 21-bit CM reference address (RAC) register loads from CM during the second word of a CYBER 170 exchange sequence. An absolute CM address forms by adding RAC to a relative address determined by the instruction. The content of the P register is added to RAC to form the program address in CM. A P-equal-to-zero condition specifies relative address 0 and, therefore, RAC. This CM location is reserved for recording error exit conditions and should not be used to store data or instructions.

FLC Register

The 21-bit CM field length (FLC) register loads from CM during the third word of a CYBER 170 exchange sequence. The FLC register defines the size of the field of the program in execution. Relative CM addresses are compared with FLC to check that the program is not going out of its allocated memory range.

EM Register

The 6-bit exit mode (EM) register loads from CM during the fourth word of a CYBER 170 exchange sequence. The EM register holds six exit mode selection bits that control individual error conditions for a program. Selected EM register bits cause the CP to error exit when the corresponding conditions occur. Any or all of the six bits can be set at one time. Cleared EM register bits allow the CP to continue, without error processing, when most of the corresponding conditions occur. Refer to the error exit tables under Error Response in chapter 3 for specific cases. The exit mode selection bits appear in the exchange package as bits 48 through 50 and 57 through 59. The bits and their corresponding conditions are:

Bit	Significance
48	Address out of range (AOR)
49	Infinite operand
50	Indefinite operand
57	Hardware error
58	Hardware error
59	Hardware error

Flag Register

The 6-bit flag register loads from CM during the fourth word of a CYBER 170 exchange sequence. The flag register holds six bits that function as control flags.

Bit	Condition
51	Hardware error bit.
52	Instruction stack (lookahead) purge flag. If set, extended purging of instruction lookahead registers is enabled. For further information, refer to Instruction Lookahead Purge Control under CP Programming in chapter 3.
53	Compare/move interrupted flag. If set, one of instructions 464 through 467 has been interrupted. The information necessary to resume operation has been saved.
54	Block copy flag. If set, block-copy instructions (011, 012) use bits 30 through 50 of X0 rather than A0 to determine the CM address. For further information, refer to the descriptions of the block-copy instructions in chapter 2.
55	Expanded addressing select flag. If set, UEM is operating in expanded addressing mode; if clear, UEM is operating in 24-bit standard addressing mode. For further information, refer to Addressing Modes under Memory Programming in chapter 3.
56	UEM enable flag. If set, UEM is available. This flag must be set to allow 011, 012, 014, and 015 instructions to access UEM.

RAE Register

The 21-bit UEM reference address (RAE) register loads from CM during the fifth word of a CYBER 170 exchange sequence. The lower six bits of this register are always zeros. An absolute UEM address forms by adding RAE to the relative address which is determined by the instruction.

FLE Register

The 24-bit UEM field length (FLE) register loads from CM during the sixth word of a CYBER 170 exchange sequence. The lower six bits of this register are always zeros. The FLE register defines the size of the field in UEM for the program in execution. Relative UEM addresses are compared with FLE.

MA Register

The 18-bit monitor address (MA) register loads from CM during the seventh word of a CYBER 170 exchange sequence. The MA register contains the absolute starting address of an exchange package which is used when executing a central exchange jump (013) instruction with the CYBER 170 monitor flag clear, or when honoring a monitor exchange jump to MA (262x) instruction with the CYBER 170 monitor flag clear. For further information, refer to CYBER 170 Exchange Jump in chapter 3.

Instruction Descriptions

2

CP Instruction Formats	2-1
Instruction Description Nomenclature	2-3
CP Operating Modes	2-4
CP Instruction Descriptions	2-4
CP Integer Arithmetic Instructions	2-5
Integer Pack/Unpack	2-6
Integer Sum/Difference	2-8
CP Branch Instructions	2-9
Branch	2-9
CP Block Copy Instructions	2-16
Block Copy	2-17
CP Shift Instructions	2-19
Left Shift	2-19
Right Shift	2-21
CP Logical Instructions	2-23
Logical Sum	2-24
Logical Difference	2-25
Logical Product	2-26
CP FP Arithmetic Instructions	2-27
Floating Sum	2-28
Floating Difference	2-31
Floating Product	2-34
Floating Divide	2-37
CP Jump Instructions	2-39
Jump	2-39
CP Exchange Jump Instructions	2-41
Exchange Jump	2-42
CP Compare/Move Instructions	2-44
Transmit	2-44
Compare/Move	2-45

This chapter contains the CYBER 170 State central processor (CP) instruction descriptions.

CP Instruction Formats

NOTE

CYBER 170 CP instructions use the rightmost 60 bits in the 64-bit word. The leftmost 4 bits are undefined. For these instructions, the most significant bit is bit 59 and the least significant bit is bit 0.

Program instruction words are divided into 15-bit fields called parcels. The first parcel (parcel 0) is the highest-order 15 bits of the 60-bit word. The second, third, and fourth parcels (parcels 1, 2, and 3) follow in order. Figure 2-1 shows possible parcel arrangements for instructions within a program instruction word.

An instruction may occupy one, two, or four parcels. This arrangement depends on the instruction format. When an instruction occupies two parcels, it must occupy two parcels within the same program word. A program word may be filled with a one-parcel pass instruction or an instruction acting as a two-parcel pass instruction. Pass instructions are used to fill a program word when necessary for placing a particular instruction in the first parcel of a program word or for avoiding a start of a two-parcel instruction in the fourth parcel of a program word. Pass instructions may also be used for branch entry points because a branch instruction destination address must begin with a new word. One-parcel pass instructions are 460xx through 463xx. Instructions 60xxx through 62xxx may be used as two-parcel pass instructions by setting the i instruction designator to zero. Refer to table 2-1 for CP instruction designators.

CP instructions 011 and 012 have special properties. They are 60-bit double instructions that must start at parcel 0. The programmer has the option of providing a branch instruction at parcels 2 and 3 in the same instruction word (to an error-handling software routine) or filling this space with pass instructions. Refer to instructions 011 and 012.

Instructions 013 and 464 through 467 are 60-bit instructions which must start at parcel 0. They ignore any information in parcels 2 and 3; these parcels are normally set to all zeros.

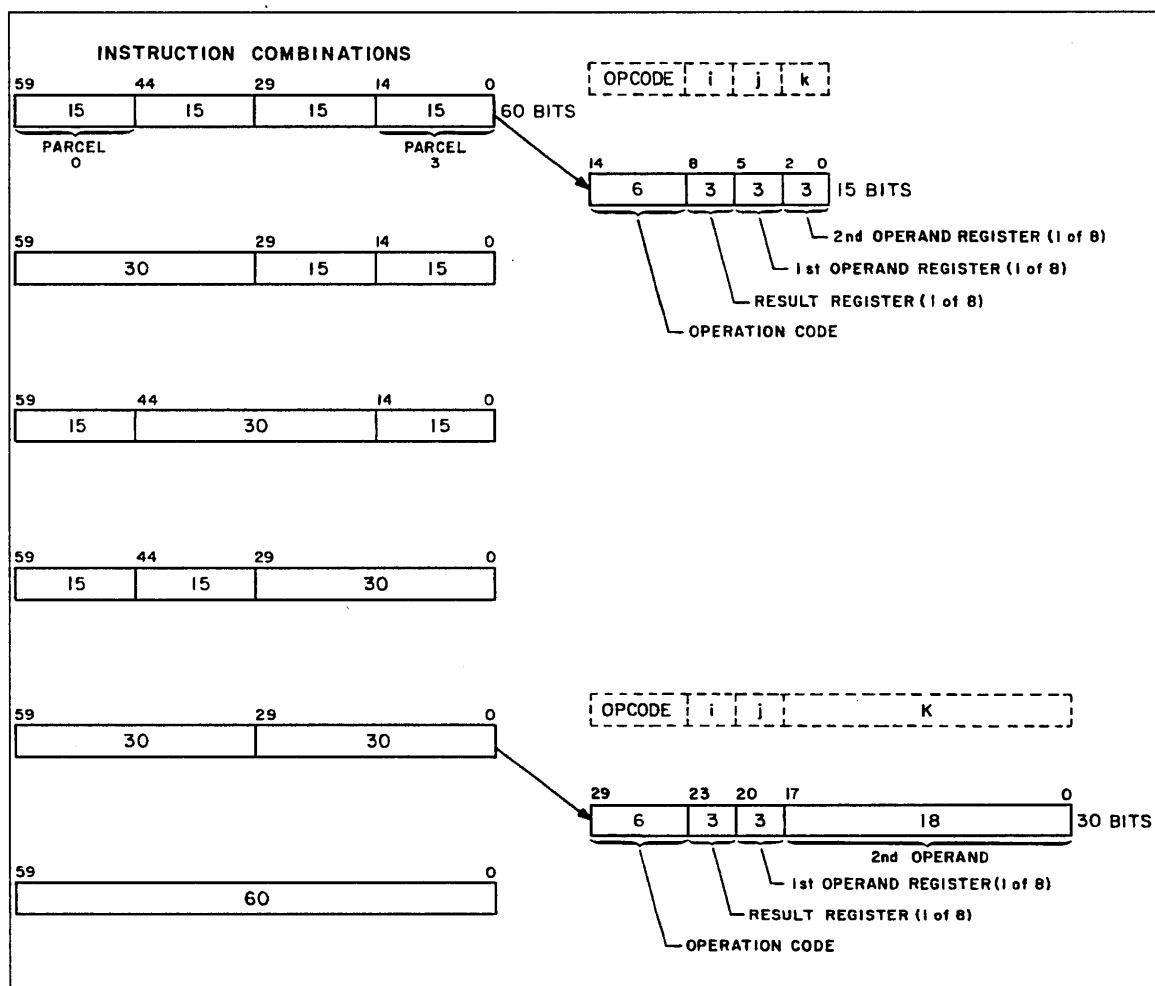


Figure 2-1. CP Instruction Parcel Arrangement

Instruction Description Nomenclature

The instruction descriptions in this chapter use the instruction designators shown in table 2-1.

Table 2-1. CP Instruction Designators

Designator	Description
Opcode	6-bit/9-bit field specifying instruction operation code
i,j,k	3-bit code specifying one of eight registers
jk	6-bit code specifying amount of shift or mask
K	18-bit operand or addresss
x	Unused designator
A	One of eight 18-bit address registers
B	One of eight 18-bit index registers; B0 is fixed and equal to zero
X	One of eight 60-bit operand registers
()	Content of the word at a central memory (CM) address
C1 ¹	Offset (character address) of the first character in the first word of the source field
C2 ¹	Character address of the first character in the first word of the result field
K1 ¹	18-bit address indicating the CM location of the first (leftmost) character of the source field
K2 ¹	18-bit address indicating the CM location of the first (leftmost) character of the result field
LL ¹	Lower 4 bits of the field length (character count) for a move or compare instruction; used with LU to specify field length
LU ¹	Upper 9 bits of the field length (character count) for indirect move instructions or the upper 3 bits for direct instructions; used with LL to specify field length.

Note:

1. Applicable to compare/move instructions only.

CP Operating Modes

The CP executes instructions in CYBER 170 job mode, CYBER 170 monitor mode, and executive state. Changes between CYBER 170 job mode and CYBER 170 monitor mode are caused by CYBER 170 exchange jumps (CP instruction 013 and PP instructions 2600, 2610, and 2620). A hardware flag called the CYBER 170 monitor flag (MF) indicates whether the CP is in CYBER 170 job mode (flag is clear) or in CYBER 170 monitor mode (flag is set).

The executive state is invisible to the applications programmer. It sets up the CYBER 170 environment during initialization, executes certain instructions, and handles hardware-detected error conditions. Hardware-caused exchanges are called error exits. Most of these can be enabled or disabled by setting or clearing bits in the CYBER 170 exchange package. For further information on CP operating modes, refer to CYBER 170 Exchange Jump, Executive State, and Error Response in chapter 3.

CP Instruction Descriptions

The CP general instructions are divided into 16 subgroups as follows:

- Integer Arithmetic
- Branch
- Block Copy
- Shift
- Logical
- Floating Point
- Jump
- Exchange Jump
- Compare/Move
- Set
- Normalize
- Pass
- Illegal Instruction
- Mask
- Pop Count
- Read Free-Running Counter

CP Integer Arithmetic Instructions

The integer arithmetic instructions (table 2-2) perform integer arithmetic on signed twos-complement words or half words in Xk or XkR. The sign bit is bit 0 for full-word integers or bit 32 for half-word integers. These instructions are explained in the following paragraphs.

Table 2-2. CP Integer Arithmetic Instructions

Opcode	Format	Instruction	Mnemonic
27	ijk	Pack (Xk) and (Bj) to Xi	PXi Bj Xk
26	ijk	Unpack (Xk) to Xi and Bj	UXi Bj Xk
36	ijk	Integer sum of (Xj) and (Xk) to Xi	IXi Xj + Xk
37	ijk	Integer difference of (Xj) and (Xk) to Xi	IXi Xj - Xk

Integer Pack/Unpack

Opcode 27ijk

Mnemonic PXi Bj, Xk

Instruction Pack (Xk) and (Bj) to Xi

Format

14	98	65	32	0
27	i	j	k	

Remarks This instruction reads the contents of the Xk and Bj registers, packs them into a single word in floating-point (FP) format, and delivers this result to the Xi register. The coefficient for the value in the Xi register is obtained from the content of the Xk register, which is treated as a signed integer. The exponent for the value in the Xi register is obtained from the content of the Bj register, which is treated as a signed integer.

The lowest-order 48 bits in the Xi register are copied directly from the lowest-order 48 bits in the Xk register. The sign bit in the Xi register is copied directly from the sign bit in the Xk register. The exponent field in the Xi register is derived from the value in the Bj register by extracting the lowest-order 11 bits in the Bj register and modifying this quantity for exponent bias and coefficient sign.

Four sample sets of operands and packed results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

(Xk)	=	0000	4500	3333	2000	0077
(Bj)	=	00	0034			
(Xi)	=	2034	4500	3333	2000	0077
(Xk)	=	0000	4500	3333	2000	0077
(Bj)	=	77	7743			
(Xi)	=	1743	4500	3333	2000	0077
(Xk)	=	7777	3277	4444	5777	7700
(Bj)	=	00	0034			
(Xi)	=	5743	3277	4444	5777	7700
(Xk)	=	7777	3277	4444	5777	7700
(Bj)	=	77	7743			
(Xi)	=	6034	3277	4444	5777	7700

This instruction converts a number in fixed-point format to FP format. For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode 26ijk**Mnemonic** UXi Bj, Xk**Instruction** Unpack (Xk) to Xi and Bj

Format 14 98 65 32 0

26	i	j	k
----	---	---	---

Remarks This instruction reads one operand from the Xk register, unpacks this word from FP format, and delivers the coefficient and exponents to the Xi and Bj registers, respectively. The 60-bit word delivered to the Xi register consists of the lowest 48 bits unaltered from the original operand plus the upper 12 bits, each equal to the original sign bit. This is a signed integer equal to the value of the coefficient in the original operand. The 18-bit quantity delivered to the Bj register is a signed integer equal to the value of the exponent in the original operand. The 11-bit exponent field in the operand is altered to remove the bias and then sign-extended to fill out the 18-bit quantity. The sign of the coefficient is removed in this process.

Four sample sets of operands and unpacked results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

(Xk)	=	2034	4500	3333	2000	0077
(Xi)	=	0000	4500	3333	2000	0077
(Bj)	=	00	0034			

(Xk)	=	1743	4500	3333	2000	0077
(Xi)	=	0000	4500	3333	2000	0077
(Bj)	=	77	7743			

(Xk)	=	5743	3277	4444	5777	7700
(Xi)	=	7777	3277	4444	5777	7700
(Bj)	=	00	0034			

(Xk)	=	6034	3277	4444	5777	7700
(Xi)	=	7777	3277	4444	5777	7700
(Bj)	=	77	7743			

This instruction converts a number from FP format to fixed-point format. For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Integer Sum/Difference

Opcode **36ijk**

Mnemonic **IXi Xj + Xk**

Instruction Integer sum of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

36	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form a 60-bit integer sum, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are signed integers. The resulting integer sum is delivered to the Xi register. Overflow is not detected.

This instruction adds integers too large for handling by 50 through 77 instructions. The instruction also merges and compares data fields during data processing.

For further information, refer to Integer Arithmetic under CP Programming in chapter 3.

Opcode **37ijk**

Mnemonic **IXi Xj - Xk**

Instruction Integer difference of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

37	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form a 60-bit integer difference, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are signed integers. The result of subtracting the quantity in the Xk register from the quantity in the Xj register is delivered to Xi. Overflow is not detected.

This instruction subtracts integers too large for handling by 50 through 77 instructions. The instruction also compares data fields during data processing.

For further information, refer to Integer Arithmetic under CP Programming in chapter 3.

CP Branch Instructions

The branch instructions (table 2-3) consist of both conditional and unconditional branch instructions. Each conditional branch instruction compares the contents of two general registers to determine whether a normal or a branch exit is taken.

Table 2-3. CP Branch Instructions

Opcode	Format	Instruction	Mnemonic
030	jK	Branch to K if (Xj) = 0	ZR
031	jK	Branch to K if (Xj) \neq 0	NZ
032	jK	Branch to K if (Xj) is positive	PL
033	jK	Branch to K if (Xj) is negative	NG
034	jK	Branch to K if (Xj) is in range	IR
035	jK	Branch to K if (Xj) is out of range	OR
036	jK	Branch to K if (Xj) is definite	DF
037	jK	Branch to K if (Xj) is indefinite	ID
04	ijK	Branch to K if (Bi) = (Bj)	EQ
05	ijK	Branch to K if (Bi) \neq (Bj)	NE
06	ijK	Branch to K if (Bi) \geq (Bj)	GE
07	ijK	Branch to K if (Bi) < (Bj)	LT

Branch

Opcode 030jK

Mnemonic ZR Xj, K

Instruction Branch to K if (Xj) = 0

Format

29	2120	1817	0
030	j	K	

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

Jump to K if: (Xj) = 0000 0000 0000 0000 0000 (positive zero)

Jump to K if: (Xj) = 7777 7777 7777 7777 7777 (negative zero)

This instruction branches on a zero result from either a fixed-point or FP operation.

Opcode 031jK**Mnemonic** NZ Xj, K**Instruction** Branch to K if (Xj) \neq 0

Format

29	2120	1817	0
031	j	K	

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

Continue if: (Xj) = 0000 0000 0000 0000 0000 (positive zero)

Continue if: (Xj) = 7777 7777 7777 7777 7777 (negative zero)

This instruction branches on a nonzero result from either a fixed-point or FP operation.

Opcode 032jK**Mnemonic** PL Xj, K**Instruction** Branch to K if (Xj) is positive

Format

29	2120	1817	0
032	j	K	

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The branch decision for this instruction is based on the value of the sign bit in the Xj register.

Jump to K if: Bit 59 of Xj = 0 (positive)

Continue if: Bit 59 of Xj = 1 (negative)

This instruction branches on a positive result from either a fixed-point or FP operation.

Opcode 033jK**Mnemonic** NG Xj, K**Instruction** Branch to K if (Xj) is negative

Format 29 2120 1817 0

033	j	K
-----	---	---

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The branch decision for this instruction is based on the value of the sign bit in the Xj register.

Jump to K if: Bit 59 of Xj = 1 (negative)
Continue if: Bit 59 of Xj = 0 (positive)

This instruction branches on a negative result from either a fixed-point or FP operation.

Opcode 034jK**Mnemonic** IR Xj, K**Instruction** Branch to K if (Xj) is in range

Format 29 2120 1817 0

034	j	K
-----	---	---

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

Continue if: (Xj) = 3777 xxxx xxxx xxxx xxxx (positive overflow)
Continue if: (Xj) = 4000 xxxx xxxx xxxx xxxx (negative overflow)

This instruction branches on an FP quantity within the FP range. The value of the coefficient is ignored in making this branch test. An underflow quantity is considered in range for purposes of this test.

Opcode 035jK**Mnemonic** OR Xj, K**Instruction** Branch to K if (Xj) is out of range

Format

29	2120	1817	0
035	j	K	

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

Jump to K if: (Xj) = 3777 xxxx xxxx xxxx xxxx (positive overflow)

Jump to K if: (Xj) = 4000 xxxx xxxx xxxx xxxx (negative overflow)

Opcode 036jK**Mnemonic** DF Xj, K**Instruction** Branch to K if (Xj) is definite

Format

29	2120	1817	0
036	j	K	

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

Continue if: (Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)

Continue if: (Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)

This instruction branches on an FP quantity that may be out of range but is still defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

Opcode 037jK**Mnemonic** ID Xj, K**Instruction** Branch to K if (Xj) is indefinite

Format 29 2120 1817 0

037	j	K
-----	---	---

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

Jump to K if: (Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)

Jump to K if: (Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)

This instruction branches on an FP quantity that is not defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

Opcode 04ijK**Mnemonic** EQ Bi, Bj, K**Instruction** Branch to K if (Bi) = (Bj)

Format 29 2423 2120 1817 0

04	i	j	K
----	---	---	---

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of the Bi and Bj registers. The branch to address K occurs only if the two quantities are identical on a bit-by-bit comparison basis. The current program sequence continues for all other cases.

This instruction branches on an index equality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

Opcode 05ijK**Mnemonic** NE Bi, Bj, K**Instruction** Branch to K if (Bi) \neq (Bj)

Format 29 24 23 21 20 18 17 0

05	i	j	K
----	---	---	---

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of the Bi and Bj registers. The program sequence continues only if the two quantities are identical on a bit-by-bit comparison basis. The branch to address K occurs for all other cases.

This instruction branches on an index inequality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

Opcode 06ijK**Mnemonic** GE Bi, Bj, K**Instruction** Branch to K if (Bi) \geq (Bj)

Format 29 24 23 21 20 18 17 0

06	i	j	K
----	---	---	---

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of the Bi and Bj registers. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is greater than or equal to the content of Bj. The current program sequence continues if the content of Bi is less than Bj.

This instruction branches on an index threshold test. A positive zero quantity is considered greater than a negative zero quantity.

Opcode **07ijK**

Mnemonic LT Bi, Bj, K

Instruction Branch to K if $(B_i) < (B_j)$

Format	29	24	23	21	20	18	17	0
	07	i	j	K				

Remarks	<p>This two-parcel instruction uses the lowest-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of the Bi and Bj registers. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is less than the content of Bj. The current program sequence continues if the content of Bi is greater than or equal to the content of Bj.</p>
----------------	---

This instruction branches on an index threshold test. A positive zero quantity is considered greater than a negative zero quantity.

CP Block Copy Instructions

The block copy instructions (table 2-4) transfer 60-bit words between fields in CM and UEM.

Table 2-4. CP Block Copy Instructions

Opcode	Format	Instruction	Mnemonic
011	jK	Block copy (Bj + K) words from UEM to CM	RE Bj+K
012	jK	Block copy (Bj + K) words from CM to UEM	WE Bj+K

Block Copy

Opcode 011jK

Mnemonic RE Bj + K

Instruction Block copy (Bj + K) words from UEM to CM

Format

59	51	47	3029	0
011	j	K	INST. FOR HALF EXIT	

Remarks This instruction copies a block of Bj plus K consecutive words from UEM to CM. The source UEM address is X0 plus RAE where the bits used depend on the setting of the expanded addressing select flag in the CYBER 170 exchange package. If the flag is clear (UEM is in standard addressing mode), the UEM address is calculated using bits 0 through 22 of X0; bits 24 through 59 are ignored. If the flag is set (UEM is in expanded addressing mode), the UEM address is calculated using bits 0 through 28 of X0; bits 30 through 59 are ignored.

The destination CM address is either A0 plus RAC, or X0 plus RAC, depending on the setting of the block copy flag in the CYBER 170 exchange package. When the block copy flag is clear, the CM address is A0 plus RAC. When the block copy flag is set, the CM address is calculated using bits 30 through 50 of X0. Bits 51 through 59 must be set to zero; results are undefined if these bits are not zero.

The operation leaves Bj, X0, and A0 unchanged. Bj and K are both signed 18-bit ones complement numbers, making it possible to transfer a maximum of 131 071 60-bit words. If Bj plus K is zero, the instruction acts as a 60-bit pass instruction.

If bit 21 or 22 of the result of X0 plus RAE is a one, zeros are transferred; and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. If execution of the 011jK instruction is interrupted, it is restarted from the beginning.

This instruction is illegal if it does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear.

In standard addressing mode, 24 bits of X0 are checked against 23 bits of FLE with bit 23 of FLE equal to zero. In expanded addressing mode, 30 bits of X0 are checked against 29 bits of FLE with bit 29 equal to zero. If the X0 bits are greater than or equal to FLE, an AOR condition is detected.

If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap in physical memory, the final contents of the destination field are undefined.

For further information, refer to Block Copy Instructions in chapter 3.

Opcode 012jK**Mnemonic** WE Bj + K**Instruction** Block copy (Bj + K) words from CM to UEM

Format

59	51	47	30	29	0
012	j	K	INST. FOR HALF EXIT		

Remarks This instruction copies a block of Bj plus K consecutive words from CM to UEM. The source CM address is either A0 plus RAC or X0 plus RAC, depending on the setting of the block copy flag in the CYBER 170 exchange package. When the block copy flag is clear, the CM address is A0 plus RAC. When the block copy flag is set, the CM address is calculated using bits 30 through 50 of X0. Bits 51 through 59 must be set to zero; results are undefined if these bits are not zero.

The destination UEM address is X0 plus RAE where the bits used depend on the setting of the expanded addressing select flag in the CYBER 170 exchange package. If the flag is clear (UEM is in standard addressing mode), the UEM address is calculated using bits 0 through 22 of X0; bits 24 through 59 are ignored. If the flag is set (UEM is in expanded addressing mode), the UEM address is calculated using bits 0 through 28 of X0; bits 30 through 59 are ignored.

The operation leaves Bj, X0, and A0 unchanged. Bj and K are both signed 18-bit ones complement numbers, making it possible to transfer a maximum of 131 071 60-bit words. If Bj plus K is zero, the instruction acts as a 60-bit pass instruction.

If bit 21 or 22 of the result of X0 plus RAE is a one, zeros are transferred; and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. If execution of the 012jK instruction is interrupted, it is restarted from the beginning.

This instruction is illegal if it does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear.

In standard addressing mode, 24 bits of X0 are checked against 23 bits of FLE with bit 23 of FLE equal to zero. In expanded addressing mode, 30 bits of X0 are checked against 29 bits of FLE with bit 29 equal to zero. If the X0 bits are greater than or equal to FLE, an AOR condition is detected.

If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap in physical memory, the final contents of the destination field are undefined.

For further information, refer to Block Copy Instructions in chapter 3.

CP Shift Instructions

The shift instructions (table 2-5) shift the Xi 60-bit word through the number of bit positions determined from a computed shift count.

Table 2-5. CP Shift Instructions

Opcode	Format	Instruction	Mnemonic
20	ijk	Left shift (Xi) by jk	LXi jk
22	ijk	Left shift (Xk) nominally (Bj) places to Xi	LXi Bj Xk
21	ijk	Right shift (Xi) by jk	AXi jk
23	ijk	Right shift (Xk) nominally (Bj) places to Xi	AXi Bj Xk

Left Shift

Opcode 20ijk

Mnemonic LXi jk

Instruction Left shift (Xi) by jk

Format 14 98 65 0

20	i	jk
----	---	----

Remarks This instruction reads one operand from Xi, shifts the 60-bit word left circularly by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

A left-circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end of the 60-bit word are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of 1 and 0 as in the original operand.

A sample computation is listed in octal notation to illustrate the operation performed.

Initial (Xi)	=	2323	6600	0000	0000	0111
jk	=		12			
Final (Xi)	=	7540	0000	0000	0022	2464

This instruction, together with instruction 21, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, use instruction 22 or 23.

Opcode **22ijk**

Mnemonic **LXi Bj, Xk**

Instruction **Left shift (Xk) nominally (Bj) places to Xi**

Format **14 98 65 32 0**

22	i	j	k
----	---	---	---

Remarks This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is left-shifted circularly the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is right-shifted with sign extension the number of bit positions designated by the value in Bj. Bj bit 17 determines the sign of Bj.

A left-circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of 1 and 0 as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced towards the lowest-order positions. The bits shifted off the lower end are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. An example of a positive shift count resulting in a left-circular shift is as follows.

```
(Xk)  =  2323  6600  0000  0000  0111
(Bj)   =    00  0012
(Xi)   =  7540  0000  0000  0022  2464
```

An example of the right shift with sign extension is as follows.

```
(Xk)  =  1327  6000  0000  3333  2422
(Bj)   =    77  7771
(Xi)   =  0013  2760  0000  0033  3324
```

If Bj bits 6 through 10 are different from Bj bit 17 and Bj bit 17 is set, the shift count is greater than 63 (decimal) places right, and a result of positive zero is returned to Xi. Bj bits 11 through 16 are not tested by this instruction.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of an FP number when the exponent has been unpacked into a B register.

Right Shift

Opcode **21ijk**

Mnemonic **AXi jk**

Instruction **Right shift (Xi) by jk**

Format **14 98 65 0**

21	i	jk
----	---	----

Remarks This instruction reads one operand from Xi, shifts the 60-bit word right with sign extension by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced toward the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. An example of a positive operand is as follows.

Initial (Xi)	=	2004	7655	0002	3400	0004
jk	=		30			
Final (Xi)	=	0000	0000	2004	7655	0002

An example of a negative operand is as follows:

Initial (Xi)	=	6000	4420	2222	0000	5643
jk	=		10			
Final (Xi)	=	7774	0011	0404	4440	0013

This instruction, together with instruction 20, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, use instruction 22 or 23.

Opcode 23ijk**Mnemonic** AXi Bj, Xk**Instruction** Right shift (Xk) nominally (Bj) places to Xi

Format

14	98	65	32	0
23	i	j	k	

Remarks This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by the content of Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is right-shifted with sign extension the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is left-shifted circularly the number of bit positions designated by the value in Bj. Bj bit 17 determines the sign of Bj.

A left-circular shift implies that the bit pattern in the 60-bit word is displaced toward the highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of 1 and 0 as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit words is displaced towards the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. The following example contains a positive shift count resulting in a right shift with sign extension.

(Xk)	=	1327	6000	0000	3333	2422
(Bj)	=	00	0006			
(Xi)	=	0013	2760	0000	0033	3324

The following example contains a negative shift count resulting in a left-circular shift.

(Xk)	=	2323	6600	0000	0000	0111
(Bj)	=	77	7765			
(Xi)	=	7540	0000	0000	0022	2464

If Bj bits 6 through 10 are different from Bj bit 17, and Bj bit 17 is clear, the shift count is greater than 63 (decimal) places right, and a result of positive zero is returned to Xi. This instruction does not test Bj bits 11 through 16.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of an FP number when the exponent has been unpacked into a B register.

CP Logical Instructions

The logical instructions (table 2-6) perform logical (Boolean) operations in the X registers.

Table 2-6. CP Logical Instructions

Opcode	Format	Instruction	Mnemonic
12	ijk	Logical sum of (Xj) and (Xk) to Xi	BXi Xj+Xk
16	ijk	Logical sum of (Xj) with complement of (Xk) to Xi	BXi -Xk+Xj
13	ijk	Logical difference of (Xj) and (Xk) to Xi	BXi Xj-Xk
17	ijk	Logical difference of (Xj) with complement of (Xk) to Xi	BXi -Xk-Xj
11	ijk	Logical product of (Xj) and (Xk) to Xi	BXi Xj*Xk
15	ijk	Logical product of (Xj) with complement of (Xk) to Xi	BXi -Xj*Xj

Logical Sum

Opcode 12ijk

Mnemonic BXi Xj + Xk

Instruction Logical sum of (Xj) and (Xk) to Xi

Format

14	98	65	32	0
12	i	j	k	

Remarks This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. The result delivered to Xi is the bit-by-bit logical sum of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj)	=	0000	7777	0123	4567	1010
(Xk)	=	0123	4567	7777	0000	1100
(Xi)	=	0123	7777	7777	4567	1110

This instruction merges portions of a 60-bit word into a composite word during data processing.

Opcode 16ijk

Mnemonic BXi -Xk + Xj

Instruction Logical sum of (Xj) with complement of (Xk) to Xi

Format

14	98	65	32	0
16	i	j	k	

Remarks This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. The result delivered to Xi is the bit-by-bit logical sum of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj)	=	0000	7777	0123	4567	1010
(Xk)	=	0123	4567	7777	0000	1100
(Xi)	=	7654	7777	0123	7777	7677

This instruction merges portions of a 60-bit word into a composite word during data processing.

Logical Difference

Opcode 13ijk

Mnemonic BXi Xj -Xk

Instruction Logical difference of (Xj) and (Xk) to Xi

Format

14	98	65	32	0
13	i	j	k	

Remarks This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. The result delivered to Xi is the bit-by-bit logical difference of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj)	=	0123	7777	0123	4567	1010
(Xk)	=	0123	4567	7777	3210	1100
(Xi)	=	0000	3210	7654	7777	0110

This instruction compares bit patterns or complements bit patterns during data processing.

Opcode 17ijk

Mnemonic BXi -Xk -Xj

Instruction Logical difference of (Xj) with complement of (Xk) to Xi

Format

14	98	65	32	0
17	i	j	k	

Remarks This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. The result delivered to Xi is the bit-by-bit logical difference of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible combinations that may occur.

(Xj)	=	0123	7777	0123	4567	1010
(Xk)	=	0123	4567	7777	3210	1100
(Xi)	=	7777	4567	0123	0000	7667

This instruction compares bit patterns or complements bit patterns during data processing.

Logical Product

Opcode 11ijk

Mnemonic BXi Xj * Xk

Instruction Logical product of (Xj) and (Xk) to Xi

Format

14		98		65		32		0
	11		i		j		k	

Remarks This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. The result delivered to Xi is the bit-by-bit logical product of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj)	=	7777	7000	0123	4567	1010
(Xk)	=	0123	4567	0077	7700	1100
(Xi)	=	0123	4000	0023	4500	1000

This instruction extracts portions of a 60-bit word during data processing.

Opcode 15ijk

Mnemonic BXi -Xk * Xj

Instruction Logical product of (Xj) with complement of (Xk) to Xi

Format

14		98		65		32		0
	15		i		j		k	

Remarks This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. The result delivered to Xi is the bit-by-bit logical product of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj)	=	7777	7000	0123	4567	1010
(Xk)	=	0123	4567	0007	7700	1100
(Xi)	=	7654	3000	0120	0067	0010

This instruction extracts portions of a 60-bit word during data processing.

CP FP Arithmetic Instructions

The FP instructions (table 2-7) perform arithmetic operations on FP numbers. (FP formats, including error indications, are described in chapter 3 of this manual.)

Table 2-7. CP FP Instructions

Opcode	Format	Instruction	Mnemonic
30	ijk	Floating sum of (Xj) and (Xk) to Xi	FXi Xj+Xk
32	ijk	Floating double-precision sum of (Xj) and (Xk) to Xi	DXi Xj+Xk
34	ijk	Round floating sum of (Xj) and (Xk) to Xi	RXi Xj+Xk
31	ijk	Floating difference of (Xj) and (Xk) to Xi	FXi Xj-Xk
33	ijk	Floating double-precision difference of (Xj) and (Xk) to Xi	DXi Xj-Xk
35	ijk	Round floating difference of (Xj) and (Xk) to Xi	RXi Xj-Xk
40	ijk	Floating product of (Xj) and (Xk) to Xi	FXi Xj*Xk
41	ijk	Round floating product of (Xj) and (Xk) to Xi	RXi Xj*Xk
42	ijk	Floating double-precision product of (Xj) and (Xk) to Xi	DXi Xj*Xk
44	ijk	Floating divide (Xj) by (Xk) to Xi	FXi Xj/Xk
45	ijk	Round floating divide (Xj) by (Xk) to Xi	RXi Xj/Xk

Floating Sum

Opcode 30ijk

Mnemonic FXi Xj + Xk

Instruction Floating sum of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

30	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form an FP sum, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in FP format and is not necessarily normalized.

The two operands are unpacked from FP format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The upper half of the result is then selected as a coefficient and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the sum is right-shifted one place, and the exponent is increased by one.

If the two operands have unlike signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form.

When the difference between the exponents is greater than 128 (decimal), the shifted sign bit is extended to the entire shifted operand. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode 32ijk**Mnemonic** DXi Xj + Xk**Instruction** Floating double-precision sum of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

32	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers; operates on them to form a double-precision, FP sum; and delivers the lower half of this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in FP format and is not necessarily normalized.

The two operands are unpacked from FP format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right-shifted by one place, and the exponent is increased by one. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode 34ijk**Mnemonic** RXi Xj + Xk**Instruction** Round floating sum of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

34	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form a rounded FP sum, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The result is delivered to Xi in FP format and is not necessarily normalized.

The round FP sum is a single-precision, FP sum with a round bit (or bits) inserted before the add operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is inserted in the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have unlike signs. The second round bit is inserted before the coefficient is shifted by the difference of the exponents. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Floating Difference

Opcode 31ijk

Mnemonic FXi Xj -Xk

Instruction Floating difference of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

31	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form an FP difference, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in FP format and is not necessarily normalized.

The two operands are unpacked from FP format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The upper half of the result is then selected and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the result is right-shifted one place, and the exponent is increased by one.

If the two operands have like signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode **33ijk****Mnemonic** **DXi Xj -Xk****Instruction** Floating double-precision difference of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

33	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers; operates on them to form a double-precision, FP difference; and delivers the lower half of this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in FP format and is not necessarily normalized.

The two operands are unpacked from FP format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right-shifted one place, and the exponent is increased by one.

Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode **35ijk**

Mnemonic **RXi Xj -Xk**

Instruction Round floating difference of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

35	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form a rounded FP difference, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in FP format and is not necessarily normalized.

The round FP difference is a single-precision, FP difference with a round bit (or bits) inserted before the subtract operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is added to the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have like signs. The second round bit is inserted before the coefficient is shifted by the difference of the exponents. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Floating Product

Opcode 40ijk

Mnemonic FXi Xj * Xk

Instruction Floating product of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

40	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form an FP product, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The result is delivered to Xi in FP format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from FP format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift is done to normalize the result coefficient. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in FP calculations where rounding of operands is not desired, such as in multiple-precision arithmetic and in calculations involving error analysis. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode 41ijk

Mnemonic RXi Xj * Xk

Instruction Round floating product of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

41	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers, operates on them to form a rounded FP product, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The result is delivered to Xi in FP format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from FP format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. A rounding bit is added to bit position 46 of this product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift is done to normalize the result coefficient. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in single-precision, FP calculations. For multiple-precision calculations, the 40 and 42 instructions must be used. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode 42ijk**Mnemonic** DXi Xj * Xk**Instruction** Floating double-precision product of (Xj) and (Xk) to Xi

Format 14 98 65 32 0

42	i	j	k
----	---	---	---

Remarks This instruction reads operands from two X registers; operates on them to form a double-precision, FP product; and delivers the lower half of this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format and are not necessarily normalized. The lower half of the double-precision product is delivered to Xi in FP format and is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from FP format. The exponents are added to determine the exponent for the result. The result exponent is exactly 48 less than the exponent for a 40 instruction. The coefficients are multiplied as signed integers to form a 96-bit integer product. The lower half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the double-precision product has only 95 significant bits, a 1-bit left shift is done to normalize the result coefficient. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most significant bit. The lower 48 bits are always read from the 96-bit product register.

This instruction is used in multiple-precision, FP calculations. This instruction also provides for integer multiplication capabilities where both operands have an exponent value of plus or minus zero, and neither coefficient has been normalized. The integer result sent to Xi is 48 bits with 60-bit sign extension. If the result exceeds 48 bits, the hardware does not detect an overflow. An overflow check can be made by executing a 40 instruction using the same two operands. If the result is nonzero, overflow is then indicated. An integer multiply operation is not intended for use with normalized operands. Infinite (3777xxx...x and 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Floating Divide

Opcode 44ijk

Mnemonic FXi Xj/Xk

Instruction Floating divide (Xj) by (Xk) to Xi

Format

14	98	65	32	0
44	i	j	k	

Remarks This instruction reads operands from two X registers, operates on them to form an FP quotient, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

The two operands are unpacked from FP format. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into FP format and delivered to Xi.

If the exponent subtraction causes an underflow or overflow, an underflow or overflow result is returned even with the occurrence of a divide fault.

If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault causes an indefinite result to be returned to Xi.

This instruction is used in FP calculations where rounding of operands is not desired. In multiple-precision division, this instruction must be followed by a multiplication of the quotient by the divisor and subtracted from the dividend to reconstruct the remainder.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

Opcode 45ijk**Mnemonic** RXi Xj/Xk**Instruction** Round floating divide (Xj) by (Xk) to Xi

Format

14	98	65	32	0
45	i	j	k	

Remarks This instruction reads operands from two X registers, operates on them to form a rounded FP quotient, and delivers this result to a third X register. The operands for this instruction are in the Xj and Xk registers. These operands are in FP format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

The two operands are unpacked from FP format in this operation. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The Xj quantity is modified by inserting a 2525...25 round pattern below the lowest-order bit of the dividend coefficient. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into FP format and delivered to Xi.

If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault occurs. A divide fault causes an indefinite result to be returned to Xi.

This instruction is used in single-precision, FP calculations where rounding of operands is desired to reduce truncation errors.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action.

For further information, refer to FP Arithmetic under CP Programming in chapter 3.

CP Jump Instructions

The jump instructions (table 2-8) allow departure from sequential instruction execution.

Table 2-8. CP Jump Instructions

Opcode	Format	Instruction	Mnemonic
010	xK	Return jump to K	RJ
02	ixK	Jump to (Bi) + K	JP

Jump

Opcode 010xK

Mnemonic RJ K

Instruction Return jump to K

Format

29	2120	1817	0
010		K	

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. This instruction writes a special word into CM at relative address K. The current program sequence then terminates by a jump to address K plus one. The word stored in memory contains a jump instruction which causes an unconditional jump to the address of this return jump instruction plus 1.

This instruction calls a subroutine and inserts execution of the subroutine between execution of this instruction word and the following instruction word. Instructions appearing after the return jump instruction in the instruction word are not executed. The called subroutine exit must be at address K. The called subroutine entrance address must be K plus 1.

This instruction stores a 60-bit word at address K in memory. The upper half of this word contains an unconditional jump (0400) instruction with an address that is equal to the current program address plus 1. The lower half of the stored word is all zeros. The octal digits in the stored word then appear as illustrated with the x field indicating the location of the current program address plus 1.

K	0400x	xxxxx	00000	00000	Subroutine exit
K + 1	yyyyy	yyyyy	yyyyy	yyyyy	Subroutine entrance

Opcode **02ixK****Mnemonic** **JP Bi + K****Instruction** **Jump to (Bi) + K**

Format 29 24 23 21 20 18 17 0

02	i		K
----	---	--	---

Remarks This two-parcel instruction uses the lowest-order 18 bits as operand K. The instruction causes the current program sequence to terminate with a jump to address Bi plus K in CM.

This instruction allows computed branch point destinations. This is the only instruction in which a computed parameter can specify a program branch destination address. All other jump instructions have preassigned destination addresses.

The quantities in Bi and operand K are added in an 18-bit ones complement mode. The result is treated as an 18-bit positive integer that specifies the beginning address in CM for the new program sequence. The remaining instructions, if any, in the instruction word do not execute.

CP Exchange Jump Instructions

The exchange jump instructions (table 2-9) exchange the current process registers (formatted as an exchange package) with another set stored in CM, and do the following:

- When executed with CP in Virtual State monitor mode, the processor switches from monitor to job mode.
- When executed in Virtual State job mode, the processor switches from job to monitor mode and the system call bit sets in the monitor condition register (MCR 10).

In either case, the P register stored in the outgoing exchange package points to the next instruction that would have executed if the exchange had not occurred.

This instruction can cause the following exception conditions.

- Environment specification error
- System call

Refer to chapter 3 for programming information.

Table 2-9. CP Exchange Jump Instructions

Opcode	Format	Instruction	Mnemonic
013	jK	Central exchange jump to (Bj) +K (CYBER 170 monitor flag set)	XJ Bj+K
013	xx	Monitor exchange jump to MA (CYBER 170 monitor flag clear)	XJ


Exchange Jump

Opcode 013jK

Mnemonic XJ Bj + K

Instruction Central exchange jump to (Bj) + K (CYBER 170 MF set)

Format

59	51	47	30	29	0
013	j	K			

Remarks This instruction must start at parcel 0. Also, a CYBER 170 exchange package must be ready at address Bj plus K or at address MA.

This instruction stores P plus 1 into the outgoing CYBER 170 exchange package in hardware and then exchanges this CYBER 170 exchange package with the CYBER 170 exchange package stored in memory. If the CYBER 170 MF is set at the beginning of the instruction, the incoming CYBER 170 exchange package starts at absolute address Bj plus K. If the CYBER 170 MF is clear at the beginning, then the j and K fields of the instruction are ignored and the incoming CYBER 170 exchange package starts at absolute address MA, which is obtained from the outgoing CYBER 170 exchange package. In either case, the CYBER 170 MF is toggled and the outgoing CYBER 170 exchange package is stored beginning at the same CM address from where the incoming CYBER 170 exchange package is obtained. Also, the jump is always to relative address P, parcel 0, from the new CYBER 170 exchange package. Refer to CYBER 170 Exchange Jump in chapter 3.

Opcode 013xx**Mnemonic** XJ**Instruction** Monitor exchange jump to MA (CYBER 170 MF clear)

Format

59	51	47	30 29	0
013	j	K		

Remarks This instruction must start at parcel 0. Also, a CYBER 170 exchange package must be ready at address Bj plus K or at address MA.

This instruction stores P plus 1 into the outgoing CYBER 170 exchange package in hardware and then exchanges this CYBER 170 exchange package with the CYBER 170 exchange package stored in memory. If the CYBER 170 MF is set at the beginning of the instruction, the incoming CYBER 170 exchange package starts at absolute address Bj plus K. If the CYBER 170 MF is clear at the beginning, then the j and K fields of the instruction are ignored and the incoming CYBER 170 exchange package starts at absolute address MA, which is obtained from the outgoing CYBER 170 exchange package. In either case, the CYBER 170 MF is toggled and the outgoing CYBER 170 exchange package is stored beginning at the same CM address from where the incoming CYBER 170 exchange package is obtained. Also, the jump is always to relative address P, parcel 0, from the new CYBER 170 exchange package. Refer to CYBER 170 Exchange Jump in chapter 3.

CP Compare/Move Instructions

The compare/move instructions (table 2-10) move characters from one CM location to another and compare fields of characters either directly or through a collate table. The transmit instructions move words from one CM register to another.

Table 2-10. CP Compare/Move Instructions

Opcode	Format	Instruction	Mnemonic
10	ijx	Transmit (Xj) to Xi	BXi Xj
14	ixk	Transmit complement of (Xk) to Xi	BXi -Xk
464	jK	Move indirect	IM
465		Move direct	DM
466		Compare collated	CC
467		Compare uncollated	CU

Transmit

Opcode 10ijx

Mnemonic BXi Xj

Instruction Transmit (Xj) to Xi

Format 14 98 65 32 0

10	i	j	
----	---	---	--

Remarks This instruction transfers a 60-bit word from Xj into Xi.

This instruction moves data from one X register to another X register. No logical function is performed on the data.

Opcode 14ixk

Mnemonic BXi -Xk

Instruction Transmit complement of (Xk) to Xi

Format 14 98 65 32 0

14	i		k
----	---	--	---

Remarks This instruction reads a 60-bit word from Xk, complements the word, and writes the result into Xi.

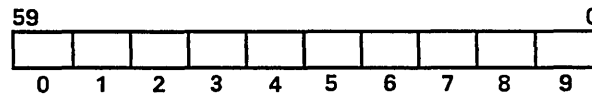
This instruction changes the sign of a fixed-point or floating-point quantity. The instruction also inverts an entire 60-bit field during data processing.

Compare/Move

The compare/move instructions (also referred to as compare/move instructions) are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of compare/move instructions.

Compare/move instructions must appear in parcel 0 or they are treated as illegal instructions.

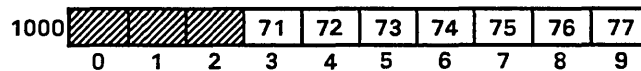
Data fields consisting of 6-bit characters may start or end with any character position (offset) of the ten 6-bit positions in each word. The character positions are designated as follows.



For move instructions, a K1 designator specifies which CM word contains the first character of the source data field, and a C1 designator specifies the character position (offset) of the first character. The K2 designator specifies the CM location in which the first character of the result data field is placed, and the C2 designator specifies the first character position. For compare instructions, both data field addresses specify source fields.

Example:

If the instruction is K1=1000 and C1=3, the first character of the source field is in position 3 of location 1000.



Therefore, the first character of the source field is 71.

An address is out of range if C1 or C2 is greater than 9, K1 plus N1 is greater than the program field length for CM (FLC), or K2 plus N2 is greater than FLC. N1 equals the number of CM references made to the source data field starting at K1, and N2 equals the number of CM references made to the result data field starting at K2. When an address-out-of-range condition occurs, the compare/move instruction is not executed.

LL is the lower 4 bits, and LU is the upper 9 bits of the field length designator in numbers of characters. The maximum length of the data fields for the move direct and the compare instructions is 127 (1778) characters. The maximum data field length for the move indirect instruction is 8191 (177778) characters. If L (LU and LL combined) is zero, the instruction becomes a pass.

For overlapping move instructions, the address of the source field (specified by K1) must be greater than the address of the result field (specified by K2) to provide proper field overlap. If K1 is less than K2, part of the source field is changed during execution. The amount of change is determined by the number of CM conflicts encountered. Overlapping fields should not contain more than 377 (octal) characters because an exchange jump interrupts any compare/move operation having a decremented field length greater than 377 (octal).

Opcode 464jK**Mnemonic** IM Bj + K**Instruction** Move indirect

Format 59 5150 4847 3029 0

464	j	K				
-----	---	---	--	--	--	--

Remarks Any instructions located in the lower two parcels of the instruction word do not execute.

Bj plus K specifies a relative address in CM for the following descriptor word.

59 5756 4847 3029 2625 22 21 18 17 0

	LU	K1	LL	C1	C2	K2
--	----	----	----	----	----	----

The descriptor word specifies the movement of the source field to the result field. The movement is from left to right through the field. Register X0 clears at the end of the execution.

Opcode 465**Mnemonic** DM**Instruction** Move direct

Format 59 5756 4847 3029 2625 22 21 18 17 0

465	LU	K1	LL	C1	C2	K2
-----	----	----	----	----	----	----

Remarks This instruction moves the source field to the result field as specified by the 60-bit instruction word. The field length is limited to a 7-bit count.

Opcode 466

Mnemonic CC

Instruction Compare collated

Format 59 5756 4847 3029 2625 2221 1817 0

466	LU	K1	LL	C1	C2	K2
-----	----	----	----	----	----	----

Remarks This instruction compares the field designated by K1,C1 with the field designated by K2,C2 as specified by the 60-bit instruction word.

The compare is from left to right through the fields until two unequal characters are found. These two characters are then collated and referenced in the collate table beginning at address A0 (table 2-11). If the table values found for the two unequal characters are equal, the compare continues until another pair of characters is unequal or until the field length is exhausted. If the table values found for the two unequal characters are unequal, X0 is set prior to instruction termination as follows.

- If field K1 is greater than field K2, set X0 to 0000 0000 0000 0000 0xxx.
- If field K1 is equal to field K2, set X0 to 0000 0000 0000 0000 0000.
- If field K1 is less than field K2, set X0 to 7777 7777 7777 7777 7yyy where yyy is the complement of xxx.

The value of the three octal numbers xxx that are stored in X0 is determined by the equation $L - N = \text{xxx}$ (L is the length of the field, and N is the number of pairs of characters that were collated equal prior to instruction termination). In other words, xxx is the number of pairs of characters not yet compared plus one.

The A0 register contains the starting word address of an 8-word, 64-character collate table (table 2-11). This table must have been previously stored in consecutive CM locations.

The collated value of a character is found by examining the collate table. The upper 3 bits of the character to be collated are added to A0 to obtain the relative address of the word containing the collated value. The lower 3 bits of the character to be collated specify the character address of the collated value.

Example:

Suppose the character under examination is an octal 63. The 6 is added to the A0 to form the word address. The 3 is used to pick the correct character from that word. The value of 63 is 63 in the collate table.

Table 2-11. Collate Table

Address	Collating Character Locations
A0	00 01 02 03 04 05 06 07
A0+1	10 11 12 13 14 15 16 17
A0+2	20 21 22 23 24 25 26 27
A0+3	30 31 32 33 34 35 36 37
A0+4	40 41 42 43 44 45 46 47
A0+5	50 51 52 53 54 55 56 57
A0+6	60 61 62 63 64 65 66 67
A0+7	70 71 72 73 74 75 76 77

Opcode 467

Mnemonic CU

Instruction Compare uncollated

Format 59 5150 4847 3029 2625 2221 1817 0

467	LU	K1	LL	C1	C2	K2
-----	----	----	----	----	----	----

Remarks This instruction is similar to the 466 instruction except that the collate table is not used. The X0 register is set when the first pair of unequal characters is encountered or when the field length is exhausted.

CP Programming	3-1
CYBER 170 Exchange Jump	3-1
Executive State	3-4
FP Arithmetic	3-4
Format	3-4
Packing	3-5
Overflow	3-7
Underflow	3-7
Indefinite	3-7
Nonstandard Operands	3-8
Normalized Numbers	3-10
Rounding	3-10
Double-Precision Results	3-10
Fixed-Point Arithmetic	3-12
Integer Arithmetic	3-13
Compare/Move Arithmetic	3-13
Instruction Lookahead Purge Control	3-14
Purge Control	3-14
Error Response	3-15
Illegal Instructions	3-23
Processor-Detected Malfunctions (EC=20 ₈)	3-24
Conditional Software Errors (EC=01 ₈ , 02 ₈ , and 04 ₈)	3-24
Monitor Condition Register Errors (EC=67 ₈)	3-24
Memory Programming	3-25
CM Layout	3-27
Addressing Modes	3-27
Direct Read/Write Instructions (014, 015, 660, 670)	3-27
Block Copy Instructions (011, 012)	3-27

This chapter contains special programming information about the central processor (CP), central memory (CM), peripheral processors (PPs), system console, real-time clock, two-port multiplexer, and maintenance channel.

CP Programming

CYBER 170 Exchange Jump

The CP operates in either CYBER 170 job mode, which is interruptable, or CYBER 170 monitor mode, which is not interruptable. A hardware flag called the CYBER 170 monitor flag (MF) indicates the mode in which the CP is executing a job.

The CP uses a CYBER 170 exchange jump operation to switch from CYBER 170 job mode to CYBER 170 monitor mode and back again. The execution of a CYBER 170 exchange jump permits the CP to send pertinent information from the operating and control registers to CM and permits CM to send new information to the same registers. The information that flows from and into the operating and control registers during a CYBER 170 exchange jump is called a CYBER 170 exchange package (figure 3-1).

The CP 013 instruction and the PP 2600, 2610, and 2620 instructions initiate a CYBER 170 exchange jump operation. A CYBER 170 exchange jump instruction starts or interrupts the CP and provides CM with the first address of a 16-word exchange package.

For the 013 instruction with MF set (CP in monitor mode), the starting address of the CYBER 170 exchange package is B_j plus K . With MF clear (CP in job mode), the address is the monitor address (MA). For the 2600 instruction, the CYBER 170 exchange package address is A plus R when bit 17 of the A register is set. When this bit is clear, the address is A . For the 2610 instruction with MF set, the instruction is a pass. With MF clear, the CYBER 170 exchange package address is A plus R when bit 17 of the A register is set. When this bit is clear, the address is A . For the 2620 instruction with MF set, the instruction is a pass. With MF clear, the CYBER 170 exchange package address is MA of the outgoing CYBER 170 exchange package.

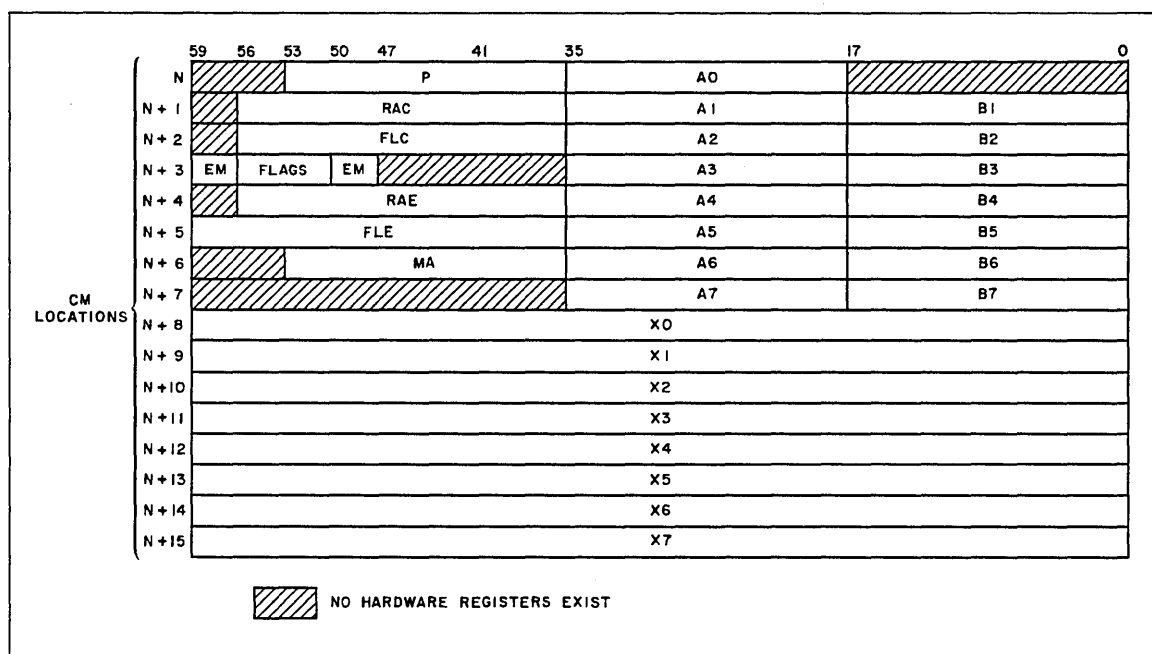


Figure 3-1. CYBER 170 Exchange Package

The CYBER 170 exchange package contains the following registers which provide information for program execution.

- 18-bit program address (P) register.
- 21-bit reference address for CM (RAC) register.
- 21-bit field length for CM (FLC) register.
- 6-bit exit mode (EM) register.
- 6-bit flag register.
- 21- or 24-bit reference address for UEM (RAE); 21 bits with lower 6 bits assumed to be zero in standard addressing mode; 24 bits right-shifted with 6 bits assumed to be zeros in expanded addressing mode.
- 21- or 24-bit field length for UEM (FLE); 21 bits in standard addressing mode and 24 bits in expanded addressing mode; lower 6 bits are assumed to be zero.
- 18-bit monitor address (MA) register.
- Initial contents of eight 60-bit X registers.
- Initial contents of eight 18-bit A registers.
- Initial contents of 18-bit B registers B1 through B7; B0 contains a constant zero.

The time that a particular CYBER 170 exchange package resides in the CP hardware registers is the execution interval. The execution interval begins with a CYBER 170 exchange jump that swaps the CYBER 170 exchange package information in CM with the information contained in the CP registers. The execution interval ends with the next CYBER 170 exchange jump.

Executive State

The executive state uses a combination of hardware, software, and microcode to handle the following items.

- System initialization.
- Compare/move instructions.
- Software errors and unimplemented instructions that occur in CYBER 170 monitor mode.
- Processor-detected hardware errors.
- Hardware integrity verification (diagnostics).

In general, executive state determines the cause of an interrupt and decides whether to return the CP to the interrupted mode, to halt the CP, or to simulate a CYBER 170 exchange and return control to CYBER 170 monitor mode. Refer to Error Response in this chapter.

FP Arithmetic

Format

FP arithmetic expresses a number in the form kB^n .

k = Coefficient

B = Base number

n = Exponent or power to which the base number is raised

B is assumed to be 2 for binary-coded quantities. In the 60-bit, FP format (figure 3-2), the binary point is considered to be to the right of the coefficient. The lower 48 bits express the integer coefficient, which is the equivalent of 15 decimal digits. The sign of the coefficient is separated from the rest of the coefficient and appears in the highest-order bit of the packed word. Negative numbers are represented in ones complement notation. The exponent is biased by complementing the exponent sign bit.

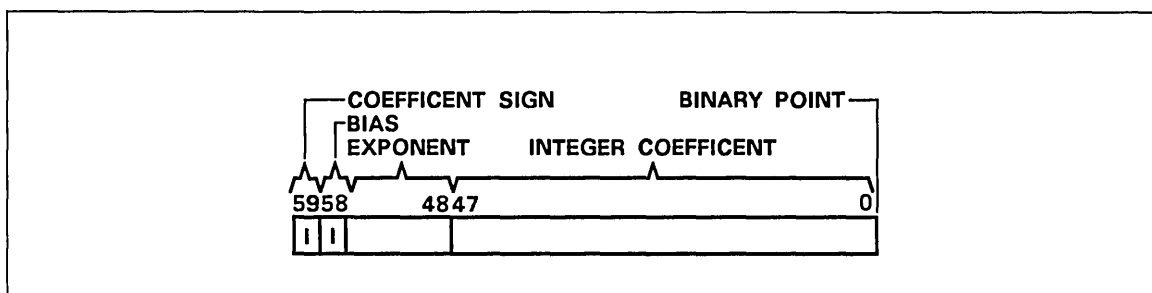


Figure 3-2. FP Format

Table 3-1 summarizes the configurations of bits 58 and 59 and the implications regarding signs of the possible combinations.

Table 3-1. Bits 58 and 59 Configurations

Bit 59	Bit 58	Coefficient Sign	Exponent Sign
0	1	Positive	Positive
0	0	Positive	Negative
1	0	Negative	Positive
1	1	Negative	Negative

Packing

Packing refers to the conversion of numbers in the form kB^n to FP format. A shortcut method of packing exponents can be derived by considering the representation of positive zero and negative zero exponents. Assuming a positive coefficient, zero exponents are packed as follows.

Positive zero exponent: 2000x,...,x

Negative zero exponent: 1777x,...,x

Since positive exponents are expressed in true form, begin with a bias of 2000 (positive zero) and add the magnitude of the exponent. The range of positive exponents is 0000 through 1777. In packed form, the range is 2000 through 3777.

When the coefficient is negative, the packed positive exponent is complemented to become 5777 through 4000.

Negative exponents are expressed in complement form by beginning with a bias of 1777 (negative zero) and then subtracting the magnitude of the exponent. The range of negative exponents is negative 0000 through negative 1777. In packed form, the range is 1777 through 0000.

When the coefficient is negative, the packed negative exponent is complemented to become 6000 through 7777.

Examples of packed and unpacked FP numbers are shown in octal notation to illustrate the packing process. Examples 1 and 2 are different forms of the integer positive 1. Example 3 is positive 100 (decimal), and example 4 is negative 100 (decimal). Examples 5 and 6 are large and small positive numbers. The unpacked values are shown as they might appear in the X and B registers prior to a pack operation.

The packed negative zero exponent is not used for normal operation. Instead, 1777 is used to indicate the special error condition of indefinite.

1.	Unpacked coefficient	0000	0000	0000	0000	0001
	Unpacked exponent	00	0000			
	Packed format	2000	0000	0000	0000	0001
2.	Unpacked coefficient	0000	4000	0000	0000	0000
	Unpacked exponent	77	7720			
	Packed format	1720	4000	0000	0000	0000
3.	Unpacked coefficient	0000	6200	0000	0000	0000
	Unpacked exponent	77	7726			
	Packed format	1726	6200	0000	0000	0000
4.	Unpacked coefficient	7777	1577	7777	7777	7777
	Unpacked exponent	77	7726			
	Packed format	6051	1577	7777	7777	7777
5.	Unpacked coefficient	0000	4771	3000	0044	7021
	Unpacked exponent	00	1363			
	Packed format	3363	4771	3000	0044	7021
6.	Unpacked coefficient	0000	6301	0277	4315	6033
	Unpacked exponent	77	6210			
	Packed format	0210	6301	0277	4315	6033

Overflow

Overflow of the FP range is indicated by an exponent value of positive 1777 (3777 or 4000 in packed form). This is the largest exponent value that can be represented in the FP format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial overflow. However, further computation using this result generates a complete overflow.

A complete overflow occurs whenever a result requires an exponent larger than positive 1777. In this case, a complete overflow value results. This result has a positive 1777 exponent and a zero coefficient. The sign of the coefficient is the same as that which generates if the result had not overflowed the FP range.

Underflow

Underflow of the FP range is indicated by an exponent value of negative 1777 (0000 or 7777 in packed form). This is the smallest exponent value that can be represented in the FP format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial underflow. Further computation using this result may be detected as a complete underflow.

A complete underflow occurs whenever a result requires an exponent smaller than negative 1777. In this case, a complete underflow value results. This result has a negative 1777 exponent and a zero coefficient. The complete underflow indicator is a word of all zeros, and it is the same as a zero word in integer format.

Indefinite

An indefinite result indicator generates whenever the calculation is unresolvable. An example is division when the divisor is 0 and the dividend is also 0. Another example is multiplication of an overflow number times an underflow number. The indefinite result indicator is a value that cannot occur in normal FP calculations. This indicator corresponds to a negative zero exponent and a zero coefficient (17770,...,0 in packed form).

Any indefinite indicator used as an operand generates an indefinite result no matter what the other operand value is. Although indefinite indicators always generate with a positive sign, they may occur as operands with a negative sign.

Nonstandard Operands

In summary, the special operand forms in octal are:

Positive overflow ($+\infty$) 3777x,...,x

Negative overflow ($-\infty$) 4000x,...,x

Positive indefinite
(+IND) 1777x,...,x

Negative indefinite
(-IND) 6000x,...,x

Positive underflow ($+0$) 0000x,...,x

Negative underflow (-0) 7777x,...,x

Tables 3-2 through 3-5 indicate the resulting forms when various combinations of underflow, overflow, and indefinite forms are used in FP operations. The designations W and N are defined as follows.

W: Any word except $+\infty$ and +IND

N: Any word except $+\infty$, +IND, and $+0$

Table 3-2. Xj Plus Xk (30, 32, and 34 Instructions)


		Xk			
		W	$+\infty$	$-\infty$	+ IND
Xj	W		$+\infty$	$-\infty$	IND
	$+\infty$	$+\infty$	$+\infty$	IND	IND
	$-\infty$	$-\infty$	IND	$-\infty$	IND
	\pm IND	IND	IND	IND	IND

Table 3-3. Xj Minus Xk (31, 33, and 35 Instructions)

		Xk			
		W	+∞	-∞	+ IND
Xj	W		-∞	+∞	IND
	+∞	+∞	IND	+∞	IND
	-∞	-∞	-∞	IND	IND
	± IND	IND	IND	IND	IND

Table 3-4. Xj Multiplied by Xk (40, 41, and 42 Instructions)

		Xk						
		+N	-N	+0	-0	+∞	-∞	+ IND
Xj	+N			0	0	+∞	-∞	IND
	-N			0	0	-∞	+∞	IND
	+0	0	0	Integer † multiply		IND	IND	IND
	-0	0	0			IND	IND	IND
	+∞	+∞	-∞	IND	IND	+∞	-∞	IND
	-∞	-∞	+∞	IND	IND	-∞	+∞	IND
	± IND	IND	IND	IND	IND	IND	IND	IND
† If both operands used in the integer multiply are normalized, an underflow results.								

Table 3-5. Xj Divided by Xk (44 and 45 Instructions)

		Xk						
		+N	-N	+0	-0	+∞	-∞	+IND
Xj	+N			+∞	-∞	0	0	IND
	-N			-∞	+∞	0	0	IND
	+0	0	0	IND	IND	0	0	IND
	-0	0	0	IND	IND	0	0	IND
	+∞	+∞	-∞	+∞	-∞	IND	IND	IND
	-∞	-∞	+∞	-∞	+∞	IND	IND	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

Normalized Numbers

A normalized FP number has as large a coefficient and as small an exponent as possible. An FP number in packed format is normalized if the coefficient sign bit is different from bit 47. This condition indicates that the coefficient has been left-shifted until bit 47 contains the most-significant bit in the coefficient; therefore, the FP number has no leading sign bits in the coefficient. The normalize instructions perform the coefficient shift. The floating-multiply and floating-divide instructions deliver normalized results when provided with normalized operands. The floating-add instructions may deliver unnormalized results even when both operands are normalized. Therefore, it is necessary to perform the normalize operation after each sequence of floating-add or floating-subtract operation if the result is to be kept in a normalized form.

Rounding

FP instructions round the results in single-precision computation. These instructions execute in the same amount of time as the unrounded versions. The operands are modified to accomplish the rounding function. The amount of bias introduced by the rounding operation varies and is affected by the coefficient value in the operands. The descriptions of the round instructions define the effects of rounding in detail.

Double-Precision Results

The FP arithmetic instructions generate double-precision results. Use of unrounded instructions allows separate recovery of upper- and lower-half results with proper exponents. Rounded instructions allow only upper-half results to be obtained. Two instructions, one single-precision and one double-precision, are required to retrieve an entire double-precision result.

To add or subtract two FP numbers, the coefficient with the smaller exponent enters the upper half of an accumulator and is right-shifted by the difference of the exponents. The other coefficient is then added into the upper half of the accumulator. The result is a double-length register (figure 3-3).

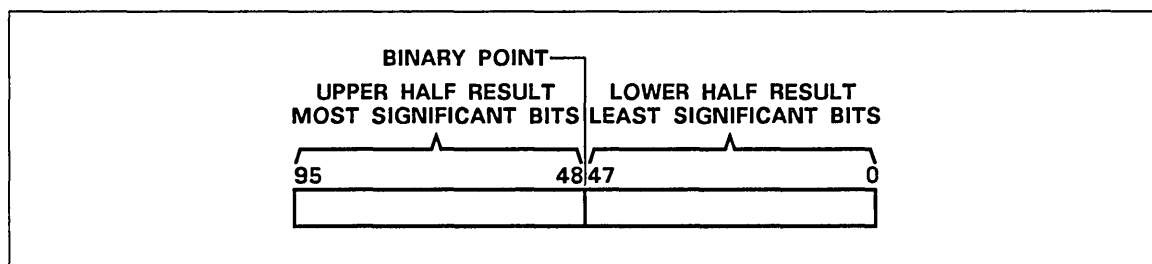


Figure 3-3. Floating-Add Result Format

If single precision is selected, the upper 48 bits of the 96-bit result and the larger exponent are returned as the result. Selecting double precision causes only the lower 48 bits of the 96-bit result and the larger exponent minus 60 (octal) to be returned as the result. The subtraction of 60 (octal) is necessary because the binary point is effectively moved from the right of bit 48 to the right of bit 0. A 96-bit product generates from two 48-bit coefficients. The result of a multiply is a double-length register (figure 3-4).

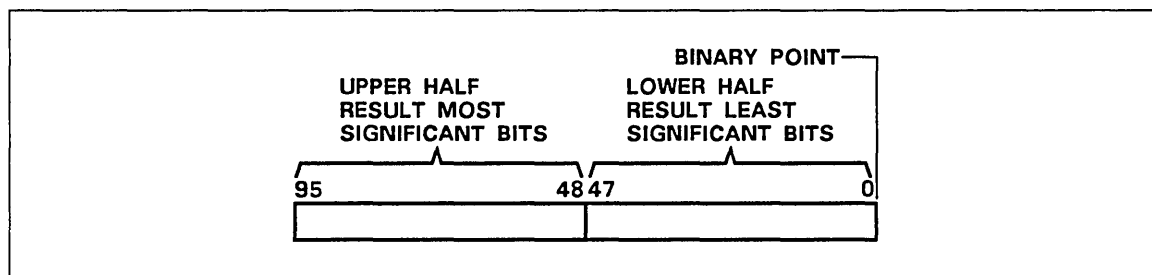


Figure 3-4. Multiply Result Format

If single precision is selected, the upper 48 bits of the product and the sum of the exponents plus 60 (octal) are returned as the result. The addition of 60 (octal) is necessary because the binary point effectively moves from the right of bit 0 to the right of bit 48 when the upper half of the 96-bit result is selected. If double precision is selected, the result is the lower 48 bits of the product and the sum of the exponents.

Fixed-Point Arithmetic

Fixed-point addition and subtraction of 60-bit numbers are handled by the long-add instructions (36 and 37). Negative numbers are represented in ones complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 59), and the binary point is to the right of the low-order bit position (bit 0).

The increment instructions (50 through 77) handle fixed-point addition and subtraction of 18-bit numbers. Negative numbers are represented in ones complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 17), and the binary point is to the right of the low-order position (bit 0).

Integer multiplication is handled as a subset operation of the floating-multiply (42) instruction. The integer multiply requires that both 47-bit integer operands have zero exponents and are not normalized. The result is 48 bits with sign extension. Normalized operands cause underflow results to be reported. If the results exceed 48 bits, overflow is not detected.

An integer divide takes several steps. For example, an integer quotient X1 equal to $X2/X3$ is produced by the following steps.

Instructions	Remarks
1. Pack X2 from X2 and B0	Pack X2
2. Pack X3 from X3 and B0	Pack X3
3. Normalize X3 in X0 and B0	Normalize X3 (divisor)
4. Normalize X2 in X2 and B0	Normalize X2 (dividend)
5. Floating quotient of X2 and X0 to Xi	Divide
6. Unpack X1 to X1 and B7	Unpack quotient
7. Shift X1 nominally left B7 places	Shift to integer position

Integer divide is handled as a subset operation of the floating-divide (45) instruction. The divide requires that both 47-bit integer operands must be in FP format, and the dividend coefficient must be less than two times the divisor coefficient. The normalize X3 instruction ensures this condition.

The normalize X3 instruction left-shifts the divisor n places ($n \geq 0$), providing a divisor exponent of negative n . The quotient exponent is then 0 minus $(-n)$ minus 48 equals n minus 48 < 0 .

After unpacking and left-shifting nominally, the negative (or zero) value in B7 right-shifts the quotient 48 minus n places, producing an integer quotient in X1. A remainder may be obtained by an integer multiply of X1 and X3 and subtracting the result from X2.

Integer Arithmetic

Integer divide packs the integers into FP format, using the pack instruction with a zero-exponent value.

In integer multiplication, a 48-bit product can be formed by using the double-precision multiply instruction. Both operands must have an exponent value of ± 0 , and the coefficients cannot both be normalized. The result is sign-extended to 60 bits and sent to an X register.

In integer division, the divisor must be normalized, but the dividend does not have to be normalized. The resulting quotient must be unpacked and the coefficient must be shifted by the amount of the unpacked exponent using the left-shift (22) instruction to obtain the integer quotient.

Compare/Move Arithmetic

The compare/move arithmetic provides multiple-character manipulation. The characters are 6 bits long. Characters can be moved from one CM location to another, and fields of characters can be compared either directly or through a collate table.

The move direct instruction moves a field of up to 127 characters from one location to another location as specified in the instruction. The move indirect instruction performs the same kind of move, but a CM reference is used to obtain the parameters. The move indirect instruction moves a field of up to 8181 characters.

The compare collated instruction compares two fields of up to 127 characters. When two characters are unequal, the characters are referenced in a collate table, and the values are compared. If those values are unequal, the field with the larger character is indicated. The compare uncollated instruction compares two fields of up to 127 characters and indicates the larger of the first character pair that is found to be unequal.

Compare/move instructions are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of compare/unit instructions.

Instruction Lookahead Purge Control

Prefetching of instructions at a branch target address by instruction lookahead hardware can lead to program failures if a program modifies its own code dynamically. Under normal conditions, the lookahead registers are purged by execution of a return jump instruction (010), UEM read instruction (011), exchange jump instruction (013), or unconditional branch instruction (02). Selecting extended purge control extends these conditions. When extended purge control is in effect, lookahead registers are also purged by execution of any conditional jump instruction (03 through 07) or any CM store instruction (50 through 57 when i equals 6 or 7). To enable extended purge control, the system sets bit 52 of the flag register in the CYBER 170 exchange package. When self-modifying code is present, it may be helpful to set extended purge control; however, the additional purging causes a degradation in execution and does not cover all cases of code modification.

Purge Control

If normal purge conditions are in effect, a store instruction that modifies a sequential instruction must modify at least P plus six words ahead to ensure execution of the modified code. In addition, a store instruction followed by a branch to a modified instruction executes the modified code only if there are at least 12 executed instructions between the store and the modified code.

If the extended purge option is selected, a store instruction can modify the next sequential instruction and be assured of executing the modified instruction. Likewise, a store instruction followed by a branch to a modified instruction always executes the modified code.

Error Response

When the CP detects or is informed of an error, it records the error. Depending on the type of error and the exit-mode selection bits set in the EM register, the program in execution may be interrupted. If the error is an illegal instruction or an address-range error on an RNI or branch, the program interruption is unconditional. For other types of errors, the exit mode selection bits determine whether or not the program is interrupted. If the exit mode selection bit is set and the corresponding condition is detected, the program is interrupted. The exit mode selection bits are contained in word N plus 3 of the exchange package. Figure 3-5 shows the format of the exit condition register at RAC. Table 3-6 describes the possible contents of the register. Tables 3-7 and 3-8 list CP error responses.

The CP has the following error conditions: illegal instructions, processor-detected malfunctions (parity errors), conditional software errors, and monitor condition register (MCR) errors.

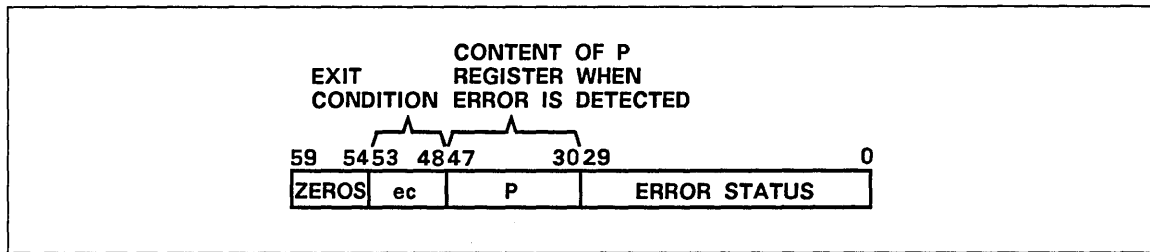


Figure 3-5. Format of Exit Condition Register at RAC

Table 3-6. Contents of Exit Condition Register at RAC

Field	Description														
ec	6-bit exit condition code: <div> <table> <tr> <th>Code (Octal)</th><th>Condition</th></tr> <tr> <td>00</td><td>Illegal instruction.</td></tr> <tr> <td>01</td><td>Address-range error (bit 48).</td></tr> <tr> <td>02</td><td>FP infinite (bit 49).</td></tr> <tr> <td>04</td><td>FP indefinite (bit 50).</td></tr> <tr> <td>20</td><td>Processor-detected malfunction (bit 52).</td></tr> <tr> <td>67</td><td>MCR errors.</td></tr> </table> </div>	Code (Octal)	Condition	00	Illegal instruction.	01	Address-range error (bit 48).	02	FP infinite (bit 49).	04	FP indefinite (bit 50).	20	Processor-detected malfunction (bit 52).	67	MCR errors.
Code (Octal)	Condition														
00	Illegal instruction.														
01	Address-range error (bit 48).														
02	FP infinite (bit 49).														
04	FP indefinite (bit 50).														
20	Processor-detected malfunction (bit 52).														
67	MCR errors.														
P	When an error exit occurs, the content of the P register may not correspond to the address of the instruction that caused the error exit. The P register may have been incremented prior to the execution of the instruction.														
ERROR STATUS	Nonzero information in bits 0 through 29 is error status for customer engineering and maintenance. MCR uses bits 0 through 15.														

Table 3-7. Error Exits in CYBER 170 Monitor Mode (MF=1)

Error Condition	Error Response Exit Mode Selected	Error Response Exit Mode Not Selected
Illegal instruction or 00 instruction. EC=00 ₈	<ol style="list-style-type: none"> 1. The instruction is not executed. 2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction. 3. Interrupt to executive state. 4. CP stops in executive state. 	<ol style="list-style-type: none"> 1. N/A (exit mode is always selected).
Exit condition bit 48 set by an incremental read with an address out of range (AOR). EC=01 ₈	<ol style="list-style-type: none"> 1. The X register is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment. 4. Interrupt to executive state. 5. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write with an address out of range (AOR). EC=01 ₈	<ol style="list-style-type: none"> 1. Block write operation; content of CM is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment. 4. Interrupt to executive state. 5. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Inhibit write, CM unchanged. 2. Continue execution.

(Continued)

Table 3-7. Error Exits in CYBER 170 Monitor Mode (MF=1) (Continued)

Error Condition	Error Response Exit Mode Selected	Error Response Exit Mode Not Selected
Exit condition bit 48 set by an RNI or branch address out of range. EC=018	<ol style="list-style-type: none"> 1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Interrupt to executive state. 4. CP stops in executive state. 	<ol style="list-style-type: none"> 1. N/A (exit mode is always selected regardless of status of EM register bit 48).
Exit condition bit 48 set on compare/move instruction. <ol style="list-style-type: none"> 1. C1 or C2 greater than 9. 2. K1 or K2 address out of range. EC=018	<ol style="list-style-type: none"> 1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Store P and exit bits (01) at RAC. 4. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Continue with next instruction.
Exit condition bit 48 set by a UEM address range check for instructions 011 and 012. EC=018	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Store P and exit bits (01) at RAC. 3. Interrupt to executive state. 4. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Exit to next 60-bit word and continue execution.

(Continued)

Table 3-7. Error Exits in CYBER 170 Monitor Mode (MF=1) (Continued)

Error Condition	Error Response Exit Mode Selected	Error Response Exit Mode Not Selected
Exit condition bit 48 set by a UEM address range check for instructions 014 and 015. EC=01 ₈	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Store P and exit condition bits (01) at RAC. P equals address of following instruction. 3. Interrupt to executive state. 4. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Exit to next parcel and continue execution.
Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition. EC=02 ₈ or 04 ₈	<ol style="list-style-type: none"> 1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following. 2. Interrupt to executive state. 3. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Continue execution.
Any hardware parity error or double SECDDED error. EC=20 ₈	<ol style="list-style-type: none"> 1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. CP stops in executive state. 	<ol style="list-style-type: none"> 1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. CP stops in executive state.

Table 3-8. Error Exits in CYBER 170 Job Mode (MF=0)

Error Condition	Error Response Exit Mode Selected	Error Response Exit Mode Not Selected
Illegal instruction or 00 instruction.	1. The instruction is not executed.	1. N/A (exit mode is always selected).
EC=00 ₈	2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction.	
	3. Exchange jump to MA and set CYBER 170 MF.	
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	1. The X register is unchanged.	1. Inhibit read, X unchanged.
	2. The A register contains the AOR address.	2. Continue execution.
EC=01 ₈	3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment.	
	4. Exchange jump to MA and set CYBER 170 MF.	
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	1. Block write operation; content of CM is unchanged.	1. Inhibit write, CM unchanged.
	2. The A register contains the AOR address.	2. Continue execution.
EC=01 ₈	3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment.	
	4. Exchange jump to MA and set CYBER 170 MF.	

(Continued)

Table 3-8. Error Exits in CYBER 170 Job Mode (MF=0) (Continued)

Error Condition	Error Response Exit Mode Selected	Error Response Exit Mode Not Selected
Exit condition bit 48 set by an RNI or branch address out of range. EC=018	<ol style="list-style-type: none"> 1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. N/A (exit mode is always selected regardless of status of EM register bit 48).
Exit condition bit 48 set on compare/move instruction. 1. C1 or C2 greater than 9. 2. K1 or K2 address out of range. EC=018	<ol style="list-style-type: none"> 1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Store P and exit bits (01) at RAC. 4. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/ writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Continue with next instruction.
Exit condition bit 48 set by a UEM address range check for instructions 011 and 012. EC=018	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Store P and exit bits (01) at RAC. 3. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Exit to next 60-bit word and continue execution.

(Continued)

Table 3-8. Error Exits in CYBER 170 Job Mode (MF=0) (Continued)

Error Condition	Error Response Exit Mode Selected	Error Response Exit Mode Not Selected
Exit condition bit 48 set by a UEM address range check for instructions 014 and 015. EC=018	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Stop CP. 3. Store P and exit condition bits (01) at location RAC. 4. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Execute instruction as a pass. 2. Exit to next parcel and continue execution.
Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition. EC=028 or 048	<ol style="list-style-type: none"> 1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following. 2. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Continue execution.
Any hardware parity error or double SECCED error. EC=208	<ol style="list-style-type: none"> 1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF. 	<ol style="list-style-type: none"> 1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.

Illegal Instructions

An instruction is illegal when it has an illegal operating code, an illegal operating parameter, or when it is positioned so that it begins in one instruction word and extends into the next instruction word. In the CYBER 170 job mode, illegal instructions cause an exchange to the CYBER 170 monitor mode. In the CYBER 170 monitor mode, illegal instructions cause a jump to executive state. The CP stops. CP illegal instructions are:

- 017
- 011, 012, 013, 464, 465, 466, and 467 if they do not begin at parcel 0
- 011, 012, 014, and 015 if the UEM enable flag in the flag register of the CYBER 170 exchange package is clear
- Any 30-bit instruction that begins at parcel 3

Processor-Detected Malfunctions (EC=20₈)

Processor detected malfunctions (PDMs) are: data parity errors, address parity errors, and double-bit errors. If the CP is in CYBER 170 job mode, a PDM causes a jump to executive state, which returns to CYBER 170 monitor mode. If the CP is in CYBER 170 monitor mode, a PDM causes a jump to executive state. The CP halts. The instruction being executed when such a fault is detected is not necessarily connected with the fault.

Conditional Software Errors (EC=01₈, 02₈, and 04₈)

Conditional software errors are caused by address-range errors and FP infinite/indefinite operands or results. A conditional software error causes action, depending on bits set in the EM field in the current CYBER 170 exchange package. If the bit reserved for use with the specific type of error is clear, the error is ignored in both CYBER 170 job and CYBER 170 monitor modes. If the bit is set and the error occurs in the CYBER 170 job mode, it causes an exchange to the CYBER 170 monitor mode.

If the bit is set and the error occurs in the CYBER 170 monitor mode, it causes an interrupt to executive state.

Monitor Condition Register Errors (EC=67₈)

MCR errors are generally program environment errors. For a listing of error causes and actions taken following an error, refer to the MCR chart in the Codes Booklet listed in About This Manual.

Memory Programming

All references to CM by the CP for instructions or read/write data are made relative to RAC. The RAC defines the lower limit of the addresses of a program in CM. The upper limit of the program addresses is defined by FLC added to RAC.

All references to UEM by the CP for instructions or read/write data are made relative to RAE. The RAE defines the lower limit of the addresses of a program/data in UEM. The upper limit of the addresses is defined by FLE added to RAE.

The field length is a number of 60-bit words established by the operating system prior to program execution. All references to CM or UEM for a program/data must be within the field established for that program.

During a CYBER 170 exchange jump, RAC and FLC are loaded into respective registers to define the CM limits of the program that is initiated by the CYBER 170 exchange jump. RAE and FLE are loaded to define the UEM limits of a program.

Figure 3-6 shows the absolute and relative memory addresses, RAC, FLC, RAE, and FLE register relationships. For a program to operate within the established limits, the following conditions must exist.

- For absolute memory addresses: $RAC \leq (RAC + P) < (RAC + FLC)$
- For relative memory addresses: $0 \leq P < FLC$

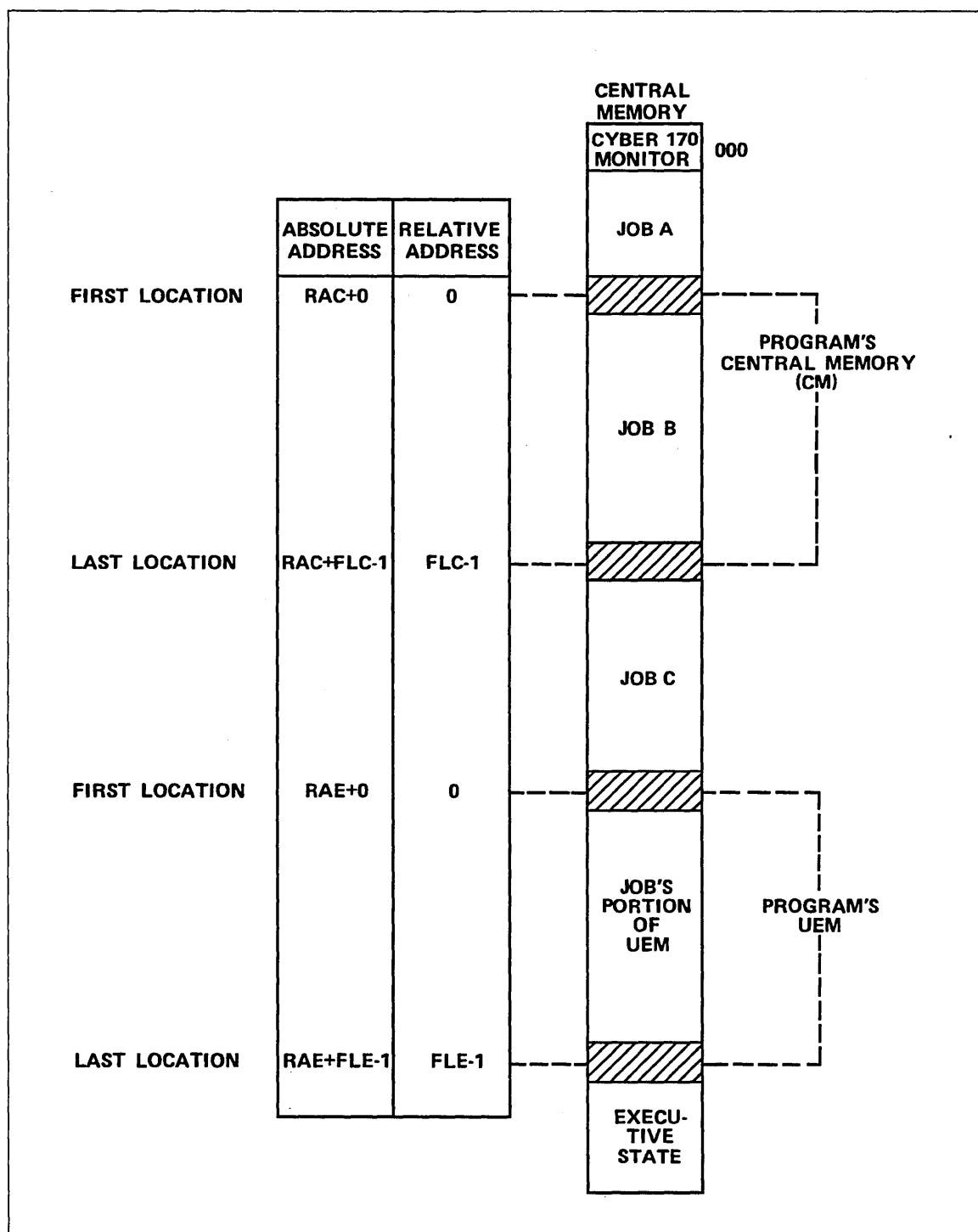


Figure 3-6. Memory Map

CM Layout

Central memory contains an area that is reserved for special software called Virtual State software. Along with the hardware and microcode, this software handles the operations of Virtual State (refer to the Virtual State Hardware Reference Manual Volumes 1 and 2 listed in About This Manual). Virtual State software is located at the higher end of memory. The remaining memory is available to the CYBER 170 State and may be allocated as CM (accessible via RAC and FLC) or as UEM (accessible via RAE and FLE and the 011, 012, 014, and 015 instructions). Refer to figure 3-7.

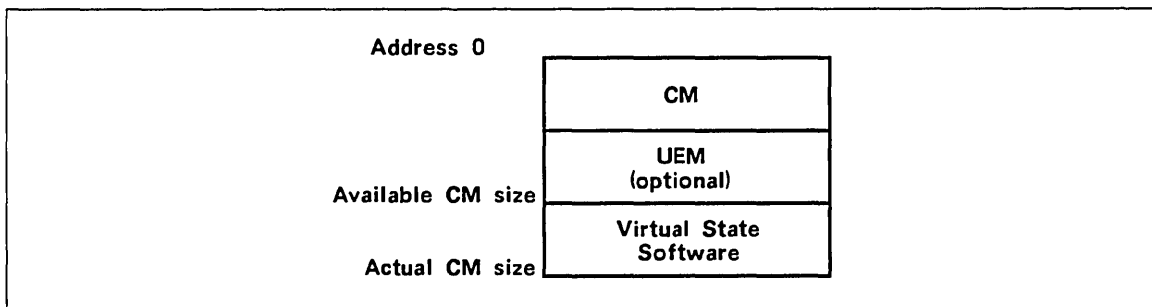


Figure 3-7. CM Layout

Addressing Modes

UEM can be used in either of two addressing modes: standard or expanded. Standard addressing mode provides addressing up to 21 bits in a 24-bit format. Expanded addressing mode provides addressing up to 24 bits in a 30-bit format. Addressing mode is determined by the expanded addressing select flag, bit 55 of word 3, in the CYBER 170 exchange package.

Direct Read/Write Instructions (014, 015, 660, 670)

These instructions transfer one 60-bit word between the selected X register and a memory location, using a 21-bit relative address. Instructions 660 and 670 use the memory address Xk (21 bits) plus RAC (21 bits) to address CM. Instructions 014 and 015 use the memory address Xk (21 bits) plus RAE (21 bits) to address UEM.

Block Copy Instructions (011, 012)

These instructions transfer up to 131 071 60-bit words between fields in CM and UEM. The UEM address is X0 plus RAE (bits 0 through 22 in standard addressing mode; bits 0 through 28 in expanded addressing mode). The CM address is A0 plus RAC (if the block copy flag is clear in the CYBER 170 exchange package) or X0 (bits 30 through 50) plus RAC (if the block copy flag is set).

The transfers occur in blocks of up to 64 words, during which other CP activities are suspended.

These instructions are 30-bit instructions that must start at parcel 0. If the UEM address has bit 21 or bit 22 set in standard addressing mode (bit 28 if in expanded addressing mode), zeros are transferred to CM and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case on a block read, the next instruction is taken from parcel 0 of the next instruction word. A transfer of all zeros can be made to central memory using the 011 instruction and setting bit 21 or 22 (or bit 28) of the address ($X0 + RAE$) when FLE is sufficiently large.

Glossary

A

A

A Register

Address register.

AC

Address control.

Address

A sequence of bits, a character, or a group of characters that identifies a network station, user, or application.

ADU

Assembly/disassembly unit in the input/output unit.

ALN

Arithmetic/logical network.

Alphabetic character

One of the following letters: A through Z, a through z.

ALU

Arithmetic logic unit.

American Standard Code for Information Interchange (ASCII)

The standard code, using a coded character set consisting of 7-bit coded characters (8-bits including parity check), used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters.

AOR

Address out-of-range.

ASCII

See American Standard Code for Information Interchange.

ASID

Active segment identifier.

AUX

Auxiliary.

B**B Register**

Index register.

BAS

Barrel and slot in the input/output unit.

BC

Base constant.

BCD

Binary-coded decimal.

BDP

Business data processor.

BN

Byte number.

BS

Binding section.

BSA

Bit significant address.

BSP

Binding section pointer.

BSR

Bit significant response.

BSS

Bus slave select.

C**Cache**

A high-speed memory, duplicating a portion of central memory, used by the central processor to speed memory access.

Carrier

A continuous frequency capable of being modulated or impressed with a signal.

CB

Circuit breaker.

CBP

Code base pointer.

CCEL

Cache corrected error log.

CDC

Control Data Corporation.

CDD

Cable configuration dependent.

CE

Customer engineer.

CEJ

Central exchange jump.

CEL

Corrected error log.

CEM

Configuration environment monitor.

Central Memory (CM)

The main memory of the CYBER 960/962 within the central processing unit cabinet. CM stores between 64 and 256 megabytes of data on 4 to 16 memory modules. CM hardware includes memory modules, a dedicated logic cage, a memory interface module, and voltage regulator modules.

Central Memory Control (CMC)

The logic element in the CPU that controls the movement of data between central memory (CMCA and CMCB) and the central processor. The CMC circuits reside on two logic modules located in the CP-0 logic cage, but accomplish memory control for both CP-0 and CP-1.

Central Processing Unit (CPU)

The main processing cabinet in the CYBER 960 Series mainframe, which includes the central processor (CP-0 and CP-1), central memory control, and central memory. The CPU cabinet has up to two central processors and between 64 and 256 megabytes of resident memory.

Central Processor (CP)

The functional processing logic within the CYBER 960/962 CPU. The first (or standard) central processor is referred to as CP-0. If a second central processor is provided with a system, it is referred to as CP-1. Each central processor resides in a separate logic cage and consists of eleven logic modules.

CF

Critical frame pointer.

CFF

Critical frame flag.

Chan

Channel.

Character

1. Any alphabetic, numeric, or special symbol that can be encoded. This term applies to the graphic characters for a input or output device and to the encoded control characters used by the terminal. Within Control Data hardware, a character is a coded byte of data, such as a 6-bit display code (NOS only) or 7-bit ASCII code. 2. (ISO) A member of a set of elements upon which agreement has been reached, and that is used for the organization, control, or representation of information. Characters may be letters, digits, punctuation marks, or other symbols. A character can be a graphic character or a control character.

CIO

See Concurrent Input/Output.

CIP

CYBER Initialization Package.

CLK

See Clock.

Clock (CLK)

1. (ISO) A device that generates periodic signals used for synchronization. 2. (ISO) Equipment that provides a time base used in a transmission system to control the timing of certain functions such as sampling and the duration of signal elements. See also Real Time.

CM

See Central Memory.

CMC

See Central Memory Control.

CMI

Control memory interface in the input/output unit.

CML

Concurrent maintenance library.

CML/VE

Concurrent maintenance library/virtual environment.

CMM

Central memory multiplexer in the input/output unit.

CMSE

Common maintenance software executive.

Coded Character Set

(ISO) A set of unambiguous rules that establish a character set and the one-to-one relationships between the characters of the set and their coded representation.

Computer room

A room that has a controlled environment that is maintained to meet the requirements of the system equipment.

Concurrent Input/Output (CIO)

An input/output unit architecture that functions with the NOS/VE operating system.

CP

See Central Processor. See also Central Processing Unit.

CP-0

See Central Processor. See also Central Processing Unit.

CP-1

See Central Processor. See also Central Processing Unit.

CPU

See Central Processing Unit.

CSF

Current stack frame.

CSSC

Customer services support center.

CST

Control store.

CST/MAC

Control store/maintenance access control.

CTI

Common test and initialization.

CYBER 960 Series (960/962) Computer Systems

Includes CYBER 960 and CYBER 962 computer systems. The CYBER 960 Series is Control Data's state-of-the-art middle- to high-range system, flanked by the 930 departmental computer on the low end and the 990 computer on the high end of the scalar performance spectrum. A full 960 system includes the mainframe (CPU, IOU, and power unit), MG set, system console, operating software, and a complement of peripherals.

C170

CYBER 170.

D**D/F**

Data/function bit.

DC

1. Direct current. 2. Debug code.

DCD

Data carrier detector.

Deadstart

The process of initializing the system by loading the operating system library programs and any of the product set from magnetic tape or disk. Deadstart recovery is reinitialization after system failure.

DEC

Dependent environment control.

Demodulation

The process of retrieving an original data signal from a modulated carrier wave.

Device Interface (DI)

The communications processor that Control Data offers as its CDCNET hardware product. Also called a CDCNET device interface.

DI

1. Debug index. 2. See Device Interface.

Digit

One of the following characters: 0 1 2 3 4 5 6 7 8 9.

DLP

Debug list pointer.

DM

Debug mask.

DMA

Direct-memory access.

DMR

Debug mask register.

Down

A status of suspended service.

DS

See Deadstart.

DSC

Display station controller.

DSP

Dynamic space pointer.

DSR

Data set ready.

DTR

Data terminal ready.

Dual CP

See Central Processor.

Dual IOU

An IOU installation containing a standard (attached) IOU and a standalone IOU.

DUE

Dependent environment control.

DVS

Diagnostic virtual system.

E**EBCDIC**

See Expanded Binary Coded Decimal Interchange Code.

EC

Environment control.

ECL

Error correction code. //PUB

Emitter-coupled logic.

ECM

Extended central memory.

ECS

Extended core storage.

EI

Environmental interface.

EIA

Electronics Industries Association.

EID

Element identifier.

EM

Error mode.

EPF

External procedure flag.

EQ

Equal.

ES

End suppression toggle (BDP edit instruction).

ESM

Extended semiconductor memory.

ESM-II

Extended semiconductor memory II.

Expanded Binary Coded Decimal Interchange Code (EBCDIC)

The set of 256 characters, each presented by eight bits, that is used with the 3270 Binary Synchronous Communications protocol.

EXT

External.

F**FCC**

Federal Communication Compliance.

FCO

Field change order.

Federal Communication Compliance (FCC)

This equipment generates, uses and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device (insert peripheral computing device if appropriate) pursuant to Subpart J of Part 15 of the FCC Rules which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

FF

See Flip-Flop.

Field-Replaceable Unit (FRU)

Equipment parts that are replaceable at the customer site are referred to as FRUs. Equipment manuals generally recommend a list of FRUs that should be stocked on site for that equipment. FRUs are identified with 8-digit part numbers that can be ordered from Control Data's World Distribution Center (WDC). See also World Distribution Center.

FIFO

See First-In/First-Out.

First-In/First-Out (FIFO).

1. This term applies to data processing services in which requests are serviced in the same order they are received. 2.(ANDIPS) A queuing technique in which the item that has been in the queue for the longest time is the next to be processed.

FL

Field length.

FLC

Central memory field length register.

FLE

Extended core storage field length register.

Flip-Flop

(ANDIPS) A circuit or device containing active elements, capable of assuming either one of two stable states at a given time. Synonymous with bistable circuit, toggle (l).

Floating-Point Base

(ISO) In a floating-point representation system, the implicit fixed positive integer base, greater than unity, that is raised to the power explicitly denoted by the exponent in the floating-point representation or represented by the characteristic in the floating-point representation and then multiplied by the fixed-point part to determine the real number represented, e.g., in the floating-point representation of the number 0.0001234, namely 0.1234-3, the implicit floating-point base is 10. Synonymous with floating-point radix.

FP

Floating-point. See Floating-Point Base.

FRC

Free-running counter.

Frequency converter

A motor generator mounted within its control cabinet.

FRU

See Field-Replaceable Unit.

FS

Fault status.

FU

Functional unit.

G**G/L**

Global/local.

Graphic Character

(ISO) A character other than a control character, that is normally represented by a graphic.

H**Hardware**

1.(ISO) Physical equipment as opposed to programs, procedures, rules, and associated documentation. 2. Electronic circuits and its housing, including cabinet, power hook-up, and cooling system.

Hdr

Header.

HEX

See Hexadecimal.

Hexadecimal

(ISO) Synonym for sexadecimal. See also Sexadecimal.

HPA/VE

Hardware performance analyzer/virtual environment.

Hz

See Hertz.

I**I/F**

See Interface.

I/O

Input/output.

IC

Integrated circuit.

ICC

Instruction completion control.

ICP

Instruction control pipe.

ID

1. Identification. 2. Identifier.

IDX

Index.

IF

Instruction fetch.

IF/IC

Instruction fetch/instruction control.

ILH

Instruction look-ahead.

Input/Output Unit (IOU)

IOU contains the peripheral processors and channels that enable operator interaction with, and peripherals access to, the central processing unit. The IOU has either NOS and NOS/VE capability (CYBER 960) or is NOS/VE only (CYBER 962). The IOU has the interface port for the system console.

INSTR

Instruction.

Interface

1. A shared boundary. An interface might be a hardware component to link two devices or it might be a portion of storage or registers accessed by two or more computer programs. 2. See data transmission interface.

International Standards Organization (ISO)

A worldwide standards group similar in function to the American National Standards Institute (ANSI). ANSI is a member of the International Standards Organization.

IOU

See Input/Output Unit.

IOU Expansion

An add-on IOU product that physically attaches to the CYBER 960 IOU. The IOU expansion is a concurrent input/output (CIO) architecture and allows for addition of up to ten peripheral processors and ten channels. The IOU expansion is the first IOU option to be added to a CYBER 960 mainframe, increasing the number of IOU cabinets to two.

IPI

Intelligent peripheral interface channel architecture in the IOU.

ISI

Intelligent standard interface channel architecture in the IOU.

ISO

See International Standards Organization.

J**JPS**

Job process state.

K**K Register**

Operation code register.

KEY

Key.

L**Large-Scale Integrated Circuit**

A chip that contains around 100 to 1000 logical gates.

LDS

Literature Distribution Services.

LED

Light-emitting diode.

LM

See Local Memory.

LMA

See Local Memory.

LOC

Local.

Local Memory (LM)

The storage area with accompanying control that provides a high-speed interface between the central processor and central memory. LM is contained by two logic modules, LMA and LMB. See also Cache.

LOCK

Lock.

Logic Cage

Hardware structure in mainframe central processing unit, input/output unit, or power unit that houses logic as memory modules or power boards.

Logic Module

A printed-circuit board with microcircuit chips.

Long Warning (LW)

The power unit generates an LW when an emergency power shutdown is to occur in approximately 1 minute if the fault is in an IOU or in approximately 2 minutes if the fault is in the power unit, CPU cabinet, or with ambient room temperature (assuming the detected fault continues).

LPID

Last processor identification.

LRN

Largest ring number.

LSB

Least significant bit.

LSI

See Large-Scale Integrated Circuit.

LW

See Long Warning.

M**MA**

Monitor address.

MAC

Maintenance access control.

Mainframe

Includes the central processing unit, input/output unit, and power unit portions of the mainframe complex. The mainframe does not include the motor-generator set or the system console.

Mainframe Complex

The hardware products that make up a central processing computer system are referred to collectively as the mainframe complex. Includes the central processing unit, input/output unit, power unit, system console, and the motor-generator set. Peripherals are not included in the mainframe complex.

MAINT

Maintenance.

MALET

Maintenance application language for equipment testing.

MCEL

Map corrected error log.

MCH

Maintenance channel.

MCR

Monitor condition register.

MCU

Maintenance control unit.

MD

Multiply/divide.

MDD

Monitor display driver.

MDF

Model-dependent flags.

MDW

Model-dependent word.

MEJ

Monitor exchange jump.

Memory Interface Module

The logical assembly interfacing central memory and central memory control. The memory interface module is physically attached to the rear of the central memory logic cage and includes circuits for data, addressing, and control.

Memory Module

The logical assembly upon which data is stored in central memory. Four memory modules constitute one memory increment or 64 megabytes of data storage. The central processing unit's central memory contains a minimum of four memory modules and a maximum of 16.

Meter

A unit of measure in the metric system that is equal to 39.3 inches.

MF

Monitor flag.

MG

See Motor-Generator Set.

Micrand

Control store word.

MMR

Monitor mask register.

Modem

(ISO) A functional unit that modulates and demodulates signals. One of the functions of a modem is to enable digital data to be transmitted over analog transmission facilities. Modem is a contraction of modulator-demodulator.

Modulation

A message signal that is impressed on a carrier signal and transmitted at another signal frequency.

Module Assembly

The circuit boards in the CP. There are 13 different modules in the CP.

Mon Cond

Monitor condition.

MOP

Micro-operator (BDP edit instruction).

MOS

Metal-oxide-semiconductor.

Motor-Generator (MG) Interface Unit

The electrical box that interfaces with the motor-generator set. Also see Motor-Generator (MG) Set.

Motor-Generator (MG) Set

A motor generator and a separate motor-generator control cabinet that converts 3-phase site utility power to 3-phase, 400-Hz power suitable for the mainframe electrical requirements. Smaller MG sets may include the MG and its MG control functions within one cabinet and be referred to as a frequency converter.

MPS

Monitor process state pointer.

MR

Maintenance register.

MSB

Most significant bit.

MSG

Message.

MSL

Maintenance software library.

Multiplexer (MUX)

Equipment that enables a site to concentrate data transmission between multiple slower-speed devices (such as, terminals and workstations) and a higher-speed channel. For example a multiplexer can concentrate data being transmitted between multiple terminals and an host computer by using a local area network.

Must

A mandatory requirement.

MUX

See Multiplexer.

N**Network Operating System (NOS)**

An operating system for the host computer. It has network capabilities for time-sharing and transmission processing in addition to local and remote batch processing. NOS controls the computation of programs submitted through remote terminals and maintains normal batch processing operations for jobs submitted locally.

Network Operating System/Virtual Environment (NOS/VE)

An operating system for the host computer. It has network capabilities for time-sharing and transmission processing in addition to local and remote batch processing. NOS/VE operates in Virtual State and controls the computation of programs submitted through remote terminals and maintains normal batch processing operations for jobs submitted locally.

NIO

See Nonconcurrent Input/Output.

Nonconcurrent Input/Output (NIO)

An input/output unit architecture that functions with NOS.

NOS

See Network Operating System.

NOS/VE

See Network Operating System/Virtual Environment.

NPA

Network performance analyzer.

NS

Negative sign toggle.

O**OCF**

On-condition flag.

ON

Occurrence number.

Opcode

Operation code.

Operand Issue (OPI)

That portion of the central processor responsible for storage and distribution of the process state register values while a process is executing. OPI is contained by logic modules OPIA, OPIB, and OP/SM.

OPI

See Operand Issue.

OPI/SM

Operand issue/segment map.

Optl

Optional.

OS

Operating system.

P**P/N**

Part number.

P Register

Program address register.

PAR

Parity.

PCB

Printed-circuit board. See Logic Module.

PDM

Processor Detected Malfunction. Any error detected by the central processor during instruction execution.

PE

Parity error.

PFA

Page frame address.

PFS

Processor fault status.

PID

Processor identifier.

PIT

Process interval timer.

PM

See Preventive Maintenance.

PMF

Performance monitoring flag.

PN

Page number.

PND

Process-not-damaged flag.

PO

Page offset.

PONR

Point of no return.

Port

The physical connection on the device interface through which data is transferred to/from the device interface. Each port is numbered (labeled) and supports a single communication line.

Power unit

Provides power to support the electrical systems (logic, environmental, and so on) of the CPU. (The IOU has its own power supply.) The power unit occupies a cabinet that attaches to the CPU.

PP

Peripheral processor.

PPM

Peripheral processor memory.

Preventive maintenance

1. (ISO) Maintenance performed specifically to prevent faults from occurring. 2. Contrast with corrective maintenance.

Primary IOU

The IOU cabinet(s) bolted to the CPU; the IOU and, if present, the IOU non-standalone expansion.

PROM

Programmable read-only memory.

PSA

Previous save area pointer.

PSF

Previous stack frame.

PSM

Page size mask.

PSWF

Page search without find.

PTA

Page table address.

PTE

Page table entry.

PTL

Page table length.

PTM

Processor test mode.

PVA

Process virtual address.

PWR

Power.

Q**Q Register**

Operand address register.

R

RAC

Central memory reference address register.

RAE

Extended core storage reference address register.

RAM

Random-access memory.

RDS

Register/data select.

Real Time

Pertaining to the processing of data by a computer in connection with another process outside the computer according to time requirements imposed by the outside process. This term is also used to describe systems operating in conversational mode and processes that can be influenced by human intervention while they are in progress.

Real-time clock

See Real Time and Clock.

REM

Remote.

RESP

Response.

Resync

Resynchronize.

RGTR

Register.

RI

Radial interface.

RMA

Real-memory address.

RN

Ring number.

RP

Ring number indicator. //PUB

Read permission.

ROM

Read-only memory.

RS-232-C

An Electrical and Electronic Industries Association (EIA) standard that describes the interface between terminals or other data terminal equipment and modems or other data communications equipment employing a serial binary interchange.

RTA

Remote technical assistance.

RTC

See Real-Time and Clock.

RTS

Request to send.

S**SCD**

System console driver.

SCSI

Small computer standard interface in IOU. The system console interfaces with the IOU through SCSI.

SCT

Special characters table.

SDE

Segment descriptor table entries.

SDT

Segment descriptor table.

SECDDED

Single error correction/double error detection.

Secondary IOU

The IOU cabinet(s) not bolted to the CPU; the standalone IOU and, if present, the standalone IOU expansion.

SEG

Segment.

Sexadecimal

1. (ISO) Characterized by a selection, choice or condition that has sixteen possible different values or states. 2. (ISO) Of a fixed-radix numeration system, having a radix of sixteen. 3. Synonymous with hexadecimal.

SFSA

Stack frame save area.

Short Warning (SW)

The power unit generates on SW when an emergency power shutdown is to occur in 2.5 seconds. Also see Emergency Power Shutdown.

Should

A recommendation that is advised but not required.

Single processor

See Central Processor.

SIT

System interval timer.

Site

The computer room and other building locations that may include one or more motor-generator sets and data media storage.

SLAVACK

Slave acknowledge.

SM

1. Segment map. 2. Symbol.

SMA

Standardized maintenance approach quick reference.

SN

Negative sign.

SPCP

System power control panel.

SPID

Segment page identifier.

SPM

System power monitor.

SPT

System page table.

SR

Select reset.

SRT

Subscript range table.

SS

Status summary.

STA

Segment table address.

Stack

An area in memory used as temporary storage for chaining calls during task or interrupt service routine execution. Task calls are chained on a user stack. Interrupt service routine calls are chained on a supervisor stack.

Stack Frame

The area within a stack that accommodates a single call.

Standalone IOU

The standalone IOU is an option that provides the CYBER 960/962 mainframe with a dual IOU configuration. The standalone IOU is installed on an island separate from the mainframe, but is linked to the CYBER 960/962 CPU with 50-ft cables. Up to ten peripheral processors (PPs) and ten channels reside in the standalone IOU cabinet.

Standalone IOU Expansion

A standalone IOU expansion product can be added to the standalone IOU, doubling the PP and channel capability of the standalone IOU configuration. The standalone IOU expansion and the IOU expansion differ only in their physical location relative to the mainframe.

STL

Segment table length.

SV

Specification value.

SVA

System virtual address.

SW

See Short Warning.

System Console

The keyboard and display screen used to monitor and control the operating system.

T**T'**

T-prime register.

TE

Trap enable.

TED

Trap-enable delay.

TEF

Trap-enable flip-flop.

TER

Terminate.

TM

Test mode.

TOS

Top of stack.

TP

Trap pointer or test point.

TPM

Two-port multiplexer.

U**UART**

See Universal Asynchronous Receiver/Transmitter.

UCR

User condition register.

UEL

Uncorrected error log.

UEM

Unified extended memory.

UMR

User mask register.

Universal Asynchronous Receiver/Transmitter (UART)

An LSI circuit for start/stop serial data transfer.

User Cond

User condition.

UTC

Utility channels.

UTP

Untranslatable pointer.

UVMID

Untranslatable virtual machine identifier.

V**V**

1. Valid bit. 2. Also see Volt.

VC

Search control code.

VL

Segment validation.

VMCL

Virtual machine capability list.

VMID

Virtual machine identifier.

Volt (V)

A measure of the work needed to move an electric charge.

W**WDC**

See World Distribution Center.

World Distribution Center (WDC)

Control Data's ordering and distribution center for spare hardware parts, software revision packages, and the documentation produced to support its product lines.

WP

Write access control (segment descriptor field).

WR

Write/read.

X**X Register**

Operand register.

XP

Execute access control.

Z**ZF**

Zero field toggle.

ZFI

Zero-fill inhibit.

Index

Index

A

A registers
 Addressing 1-4
 Clear 3-1
 CP 1-4
 CYBER 170 exchange package 3-3
 Operating 1-4
 Absolute CM address formation 1-5; 3-25
 Bj plus K 2-42
 MA 2-42
 RAC register 1-5
 Absolute UEM address formation 1-7
 Access, CM 1-1
 Access, UEM 1-6
 Add instructions 2-28, 29, 30; 3-10, 11
 Sum/Difference 2-8, 24, 25, 28
 Address
 A0 2-48
 Bi 2-40
 Bj 2-42
 Character 2-3, 47
 CM 1-4; 2-3, 17, 18, 40; 3-1, 3
 Data field 2-45
 Exchange package 1-7; 3-1
 K 2-9, 17, 18, 39
 MA 1-7; 2-42, 43; 3-1, 3
 Monitor 1-7; 3-1, 3
 P 2-42, 43; 3-3
 Parity errors 3-24
 Relative 1-4, 7; 2-47; 3-25
 UEM 1-6; 2-17, 18; 3-3, 27, 28
 Word 2-47
 Xk 3-27
 Address out of range (AOR) 1-6; 2-17, 18, 45; 3-17, 20
 Address range 2-17, 18
 Address registers (see A Registers)
 Addressing
 Absolute 1-5, 7; 3-25
 Relative address formation 1-5
 Addressing mode
 Expanded 1-6; 2-17, 18; 3-3, 27
 Standard 1-6; 2-17, 18; 3-3, 27
 Arithmetic, integer 2-4; 3-13
 CP 2-5
 Fixed point 2-9, 10
 FP 2-6, 9, 10
 Instructions 2-5

B

B registers
 CP 1-4
 CYBER 170 exchange package 3-3
 Exponent in 2-20, 22; 3-6, 7

Indexing 1-4
 Operating 1-4
 Support 1-4
 Base number 3-4
 Block copy flag 1-6; 2-4; 3-27
 Block copy instructions
 From CM instruction 2-16, 18; 3-27
 From UEM instruction 2-16, 17; 3-27
 Sequence 2-16
 Boolean instruction (logical) 2-23
 Branch destination address 2-1
 Branch instructions
 CP 2-9
 Definite 2-12
 Equal to zero 2-9
 In range 2-11
 Indefinite 2-13
 Negative 2-11
 Not equal to zero 2-10
 Out of range 2-12; 3-18, 20
 Positive 2-10
 Register greater than or equal 2-14
 Register less than 2-15
 Registers equal 2-13
 Registers not equal 2-14
 Branch target address 3-14

C

Central exchange jump instruction (see CYBER 170 Exchange Jump)
 Central memory (see CM)
 Central processor (see CP)
 Channel, maintenance (see Maintenance Channel)
 Character address designator 2-3, 45
 Character collation 2-47, 48; 3-13
 Character count 2-3
 Character manipulation 3-13
 Character size 2-3
 Characteristics, 1-1
 Chassis configuration 1-2
 Clear block copy flag 1-6; 2-17, 18; 3-27
 Clear CYBER 170 monitor flag (exchange package) 1-7; 2-4, 43; 3-1, 23
 Clear EM register bits 1-6
 Clear expanded addressing select flag 1-6; 2-17, 18
 CM
 Address format 1-4, 5, 6, 7
 Address formation 1-4, 5, 6, 7
 Block copy instructions 1-6
 CYBER 960 CP 1-1
 Description 1-1
 Destination address 2-17
 Layout 1-2
 Map 3-26

- Program address 1-5
- Programming 1-4; 3-1
- Reference address register 1-5
- Word 1-4; 2-45
- Collate table for compare/move instruction 2-47, 48; 3-13
- Compare collated instruction 2-47; 3-13
- Compare/move arithmetic 3-13
- Compare/move instruction 2-44
- Compare/move instructions
 - As error exits (table) 3-17
 - Compare collated 2-47
 - Compare uncollated 2-48
 - Designators within instructions 2-47
 - Move direct 2-46; 3-13
 - Table 2-44
- Compare/move interrupted flag 1-6
- Compare uncollated instruction 2-48
- Complement, instructions using 2-5, 23, 44; 3-4, 12
- Conditional software errors 3-4, 15, 24
- Configuration, system 1-1
- Control flags 1-6
- Control registers 3-1
- Conversion
 - Fixed-point to floating-point 2-6
 - Floating-point to fixed-point 2-7
- CP
 - Addressing section 1-5
 - Description 1-1
 - Execution section 1-4
 - Functional characteristics 1-1
 - Instruction section 1-6
 - Programming 1-4; 3-1
 - Registers 1-4
- CYBER 170 exchange jump 1-3, 5, 7; 2-4, 41; 3-1, 25
 - CP 2-41; 3-1
 - Instruction 013 1-7; 2-1, 4, 41; 3-1, 14, 23
- CYBER 170 exchange package 1-7; 2-4, 41, 42, 43; 3-1, 2, 3, 23, 27
- CYBER 170 exchange sequence 1-5, 7
- CYBER 170 job mode 2-4; 3-1, 23
- CYBER 170 monitor flag 1-7; 2-4, 43; 3-1
- CYBER 170 monitor mode 2-4; 3-1, 4, 20, 21, 22
- Cycle times 1-3

D

- Data fields 2-8, 45
- Data parity error 3-24
- Data shift 2-20, 21, 44
- Direct read/write instructions 3-27
- Double-bit errors 3-19, 24
- Double-precision results 2-27, 29, 32, 34, 35, 36; 3-10

E

- EM register 1-6; 3-3, 15, 18, 21
- Environment specification error 2-41
- Error conditions 1-5, 6; 2-4; 3-4, 6, 15, 24
- Error exit 2-4; 3-17, 20
- Error exit condition codes 3-15, 16
- Error handling 2-1; 3-15, 24
- Error processing 1-5
- Error response 3-15, 24
- Exchange jump instruction 1-7
- Exchange jump (see CYBER 170 Exchange Jump)
- Exchange package (see CYBER 170 Exchange Package)
- Exchange sequence (see CYBER 170 Exchange Sequence)
- Execution interval 1-3
- Execution section 1-4
- Exit condition codes 3-15
- Exit condition register 3-15
- Exit mode 2-28; 3-3
- Exit mode register (see EM Register)
- Exit mode selection bits 2-28; 3-15
- Expanded addressing mode 1-6; 2-17, 18; 3-3, 27
- Expanded addressing select flag 2-17; 3-27
- Extended purge control (see Instruction Lookahead Purge Control)

F

- Field length for CM register (see FLC Register)
- Field length for UEM register (see FLE Register)
- Firmware or control failure 3-14
- Fixed-point arithmetic 3-12
- Fixed-point to floating-point conversion 2-6; 3-12
- Flag register 3-3, 23
- FLC register 1-5; 2-45; 3-3
- FLE register 1-7; 3-3
- Floating-add instruction 2-28; 3-10
- Floating divide instruction 2-37
- Floating double-precision difference instruction 2-32
- Floating double-precision product instruction 2-36
- Floating double-precision sum instruction 2-29
- Floating-multiply instruction sequence 2-2
- Floating-point arithmetic 2-27, 28, 29; 3-4
 - Double-precision 2-36; 3-10
 - Format 3-4
 - Indefinite 3-7
 - Nonstandard operands 3-8

- Normalized numbers 3-10
- Overflow 3-7
- Packing 3-5
- Rounding 3-10
- Underflow 3-7
- Floating-point difference instruction 2-31, 32
- Floating-point sum instruction 2-28
- Floating-point to fixed-point conversion 2-7
- Floating product instruction 2-34, 35
- Functional characteristics 1-1

G

- Glossary A-1

H

- Hardware errors 2-4; 3-4, 19, 22

I

- ILH, see Instruction lookahead
- Illegal instructions 2-17, 45; 3-15, 23
- Indefinite condition 3-19, 22
- Indefinite, floating-point 3-7
- Indefinite operand 3-24
- Index registers (see B Registers)
- Infinite operand 3-24
- Input/output unit (see IOU)
- Instruction lookahead 3-14
- Instruction lookahead purge control 3-14
- Instruction prefetch (see Instruction Lookahead)
- Instruction purge flag 3-14
- Instruction section 1-6
- Instructions, CP
 - Block copy from CM 2-16; 3-27
 - Block copy from UEM 2-16; 3-27
 - Branch 2-9
 - Central exchange jump 2-41; 3-1
 - Compare collated 2-47; 3-13
 - Compare/move 2-44
 - Compare uncollated 2-48
 - Direct read/write of CM 3-27
 - Error exit to MA 2-4; 3-17
 - Floating divide 2-37
 - Floating double-precision difference 2-32
 - Floating double-precision product 2-36
 - Floating double-precision sum 2-29
 - Floating-point difference 2-31
 - Floating-point sum 2-28
 - Floating product 2-34, 35
 - Illegal 2-17; 3-15
 - Integer difference 2-8
 - Integer sum 2-8
 - Left shift 2-19, 20
 - Logical difference 2-25

- Logical product 2-26
- Logical sum 2-24
- Monitor exchange jump 2-43
- Move direct 2-44, 46
- Move indirect 2-44, 46
- Normalize 3-10
- Pack 2-6
- Read word from CM 3-27
- Read word from UEM 3-27
- Right shift 2-21, 22
- Round floating difference 2-33
- Round floating divide 2-38
- Round floating product 2-35
- Round floating sum 2-30
- Round normalize 3-10
- Unpack 2-6
- Write one word to UEM 3-27
- Write word to CM 3-27
- Integer arithmetic 2-5
- Integer difference instruction 2-6
- Integer sum instruction 2-6
- IOU 1-1
 - Expansion 1-1
 - Standalone 1-1

J

- Job mode (see CYBER 170 Job Mode)
- Jump instructions 2-39

L

- Least significant bit 2-1
- Left circular shift 2-19
- Left shift instruction 2-19, 20
- Logical difference instruction 2-23, 25
- Logical product instruction 2-23, 26
- Logical sum instruction 2-23, 24
- Long-add instructions 3-12
- Lookahead, instruction 1-6
- Lookahead, purge control 3-14
- Lookahead purge flag 1-6; 3-14

M

- MA register 1-7
- Mainframe configuration (see Configuration, System)
- Maintenance channel 3-1
- Masking word 2-3
- MCH (see Maintenance Channel)
- MF (see CYBER 170 monitor flag)
- Microcode 3-4
- Mode selection bits 3-15
- Modes of operation 2-4
- Monitor address register (see MA Register)
- Monitor exchange jump instruction 2-4
- Monitor flag (see CYBER 170 Monitor Flag)

Monitor mode (see CYBER 170 Monitor Mode)

Most significant bit 2-1

Move direct instruction 2-44, 46; 3-13

Move indirect instruction 2-44, 46; 3-13

N

Nonstandard operands 3-8

Normal jump instruction sequence 2-39

Normalize instruction 3-10, 12

Normalize operations 3-10

Normalized numbers

Fixed-point 3-12

Floating-point 3-10, 12

O

Offset designator for compare/move 2-45

Ones complement

addition/subtraction 2-17

Opcode designator 2-3

Operand designator 2-3

Operand registers 2-8

Operating modes 2-4

Operating registers 1-4

Operation code 2-3

Overflow 3-7

Fixed-point 3-12

Floating-point 3-7

P

P register 1-3; 2-41; 3-16

Pack instruction 2-6

Pack/unpack instructions 2-6

Packing numbers 3-5

Parcels 2-1

Parity errors 3-15

Partial overflow 3-7

Partial underflow 3-7

Pass instruction 2-1, 17

Pass instructions 2-1

Peripheral processor (see PPs)

PP instructions 2-4

PPs 1-1; 3-1

Prefetch of branch target address 3-14

Program address register (see P Register)

Program mode 2-4

Programming 1-4; 3-1

Programming information 3-1

Publication index (see System Publication Index)

Purge control 1-6; 3-14

R

RAC register 1-5; 2-17; 3-3, 15

RAE register 1-7; 2-17; 3-3

Read word from CM instruction 3-27

Read word from UEM instruction 3-27

Reference address for CM register (see RAC Register)

Reference address for UEM register (see RAE Register)

Registers

A 1-4; 3-1, 3

B 1-4; 2-20; 3-6

EM 1-6; 3-3, 15

FLC 1-5; 2-45; 3-3

FLE 1-7; 3-3

MA 1-7

P 1-3; 2-41; 3-16

RAC 1-5; 2-17; 3-3, 15

RAE 1-7; 2-17; 3-3

X 1-4; 2-8; 3-3, 13

Relative address 1-4, 7; 3-25

Return jump instruction 2-39

Right shift instruction 2-19, 21

Round floating difference instruction 2-27, 33

Round floating divide instruction 2-27, 38

Round floating product instruction 2-27, 35

Round floating sum instruction 2-27, 30

Round normalize instruction 2-35

Rounding, floating-point 3-10

Rounding operation 3-10

S

SECEDED errors 3-19, 22

Shift designator 2-3

Shift instruction 2-19

Single-precision 2-30; 3-10

Software errors, 3-4, 15

Standard addressing mode 3-3

Store instruction 3-14

Subtract instruction 2-8

Support registers 1-2, 5

T

Target address 3-14

Timing considerations 1-3

Transfer block copy 2-16

Transfer word from X register to another X register 2-44

Transfer 60-bit words 2-17

Transmit complement instruction 2-44

Transmit word instruction 2-44

Two-port multiplexer interface 3-1

U**UEM**

- Access 1-6
- Block copy instructions 2-16, 17, 18
- Description 1-6
- Direct read/write instructions 3-27
- Enable flag 1-6
- Field length register (see FLE Register)
- Read one word from 3-27
- Reference address register (see RAE Register)
- Write one word to 3-27

Unconditional jump instruction 2-39

Underflow, floating-point 3-7

Unified extended memory (see UEM)

Unpack instruction 2-6

V

Virtual State 1-1; 2-41; 3-27

W

Write word to CM instruction 3-27

Write word to UEM instruction 3-27

X

X registers 1-4; 2-8; 3-3, 13

Please fold on dotted line;
seal edges with tape only.

FOLD

FOLD

FOLD

BUSINESS REPLY MAIL

First-Class Mail Permit No. 8241 Minneapolis, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA

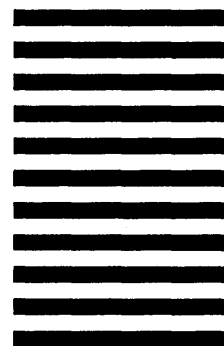
Technical Publications

ARH219

4201 N. Lexington Avenue

Arden Hills, MN 55126-9983

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



We would like your comments on this manual to help us improve it. Please take a few minutes to fill out this form.

Who are you?

- ☐ Student
☐ Instructor
☐ Customer Engineer (CE)
☐ Engineer-in-Charge (EIC)
☐ Technical support
☐ Other _____

How do you use this manual?

- ☐ To learn the product or system
☐ As a training document
☐ To maintain the product or system
☐ To troubleshoot the product or system
☐ For installation and checkout
☐ For quick look-up
☐ Other _____

How do you like this manual? Answer the questions that apply.

- | Yes | Somewhat | No | |
|--------------------------|--------------------------|--------------------------|---|
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Does it tell you what you need to know about the topic? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Is the technical information accurate? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Is it easy to understand? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Is the order of topics logical? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Can you easily find what you want? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Are there enough examples? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Are the examples helpful? (<input type="checkbox"/> Too simple? <input type="checkbox"/> Too complex?) |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Do the illustrations help you? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Is the manual easy to read (print size, page layout, and so on)? |
| <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Do you use this manual frequently? |

Comments? If applicable, note page and paragraph. Use other side if needed.

Check here if you want a reply: ☐

Name _____

Company _____

Address _____

Date _____

Phone _____

Please send program listing and output if applicable to your comment.

Central Processor Instruction Index

AX	2-21	IX	2-8
AX	2-22	JP	2-40
BX	2-24	LT	2-15
BX	2-25	LX	2-19
BX	2-26	LX	2-20
BX	2-44	NE	2-14
CC	2-47	NG	2-11
CU	2-48	NZ	2-10
DF	2-12	OR	2-12
DM	2-46	PL	2-10
DX	2-29	PX	2-6
DX	2-32	RE	2-17
DX	2-36	RJ	2-39
EQ	2-13	RX	2-30
FX	2-28	RX	2-33
FX	2-31	RX	2-35
FX	2-34	RX	2-38
FX	2-37	UX	2-7
GE	2-14	WE	2-18
ID	2-13	XJ	2-42
IM	2-46	XJ	2-43
IR	2-11	ZR	2-9
IX	2-8		

